LoNAS: Elastic Low-Rank Adapters for Efficient Large Language Models

J. Pablo Muñoz[1*], Jinjie Yuan[2*], Yi Zheng[2], Nilesh Jain[1]
[1]Intel Labs, Santa Clara, CA, USA
[2]Intel Corporation, Beijing, CN
{pablo.munoz, jinjie.yuan, yi.zheng, nilesh.jain}@intel.com

Abstract

Large Language Models (LLMs) continue to grow, reaching hundreds of billions of parameters and making it challenging for Deep Learning practitioners with resource-constrained systems to use them, e.g., fine-tuning these models for a downstream task of their interest. Adapters, such as low-rank adapters (LoRA), have been proposed to reduce the number of trainable parameters in a model, reducing memory requirements and enabling smaller systems to fine-tune these models. Orthogonal to this work, Neural Architecture Search (NAS) has been used to discover compressed and more efficient architectures without sacrificing performance compared to similar base models. This paper introduces a novel approach, LoNAS, to use NAS on language models by exploring a search space of elastic low-rank adapters while reducing memory and compute requirements of full-scale NAS, resulting in high-performing compressed models obtained from weight-sharing super-networks. Compared to models fine-tuned with LoRA, these models contain fewer total parameters, reducing the inference time with only minor decreases in accuracy and, in some cases, even improving accuracy. We discuss the limitations of LoNAS and share observations for the research community regarding its generalization capabilities, which have motivated our follow-up work.

Keywords: Language Models, Neural Architecture Search, Parameter-Efficient Fine-Tuning

1. Introduction and Motivation

The emergence of foundation models (Bommasani et al., 2021), i.e., large pre-trained models that have motivated a paradigm shift in which users focus on their adaptation and fine-tuning to a task/dataset of interest (Raffel et al., 2020a), has had a significant impact in many domains, including Natural Language Processing (NLP) and Computer Vision (CV). However, the latest advances in these large models come with a price in the form of a significant increase in their number of trainable parameters, e.g., the Pathways Language Model (PaLM) with 540 billion parameters (Chowdhery et al., 2022). These models require substantial resources for their training and inference stages. Researchers have created alternative versions of large models with fewer parameters to enable their use in more constrained environments. For instance, LLaMA has model versions with 7, 13, 33, and 65 billion parameters (Touvron et al., 2023), reducing the requirements to experiment with these models. However, the large number of parameters and the significant demand for computational resources limit many NLP practitioners from benefiting from LLMs.

To address some of the challenges and permit the use of these large models, researchers have developed Parameter-Efficient Fine-Tuning (PEFT) methods that enable the adaptation of these large models to custom datasets and tasks without having to alter any of the trainable parameters of the original pre-trained model, but only update the parameters of the inserted adapters (more details in Section 2). However, efficient fine-tuning is only one of the challenges encountered when working with large models. If their applications require deployment in resource-constrained environments, e.g., devices at the edge, many techniques for model compression have to be explored, e.g., pruning and quantization.

Orthogonal to developing sophisticated PEFT adapters, Neural Architecture Search (NAS) techniques have continued evolving, improving their efficiency and reducing the cost of discovering high-performing architectures. However, performing NAS on a large model is still an expensive endeavor. This paper attempts to address this challenge. It proposes a framework for applying NAS techniques to the trainable parameters of the PEFT adapters and keeping the weights frozen in the original large model while allowing them to be pruned based on the decisions made in the adapters' search space. The proposed framework consistently manages the dependencies between the changes made to the adapters and the corresponding frozen weights. In the following sections, we discuss the following contributions:

1. A novel framework, LoNAS, for applying weight-sharing NAS on a search space composed of configurations of PEFT algorithms. We demonstrate LoNAS using elastic low-rank (LoRA) adapters and discuss its limitations.

2. LoNAS compresses language models, result-

* Co-first authors.

LREC-COLING 2024, pages 10760–10776

ing in Pareto frontiers of more efficient model variants. These models save memory and computation, allowing for faster inference and increasing the range of devices to deploy these models.

3. Extensive experiments are conducted to study the performance and limitations of LoNAS on various datasets and models.

2. Related Work

Transformers (Vaswani et al., 2017) are the foundation of many recent large language models (LLMs) that have achieved significant performance in various tasks. Given an input $\boldsymbol{X}$, the scaled dot-product attention operator (Equation 1) in a Transformer block produces linear projections using weight matrices $\boldsymbol{W}^Q$, $\boldsymbol{W}^K$, and $\boldsymbol{W}^V$, i.e., $\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}^Q$, $\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}^K$, and $\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}^V$. In this formulation, a scaling factor, $\sqrt{d_k}$, prevents the saturation of the *softmax* function. Formally,

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \quad (1)$$

where multiple of these attention heads are concatenated to attend differently and in parallel to the input sequence of the Transformer block.

A second component of the Transformer block is a feed-forward network (FFN) that takes the output of the attention layers and transforms it before passing it to the next layer. Other elements often present in these blocks include residual connections and layer normalization. The training stage of Transformer-based models is highly parallelizable, while generative inference is not (Pope et al., 2022), presenting opportunities for new techniques that make these models more efficient at inference time. In this work, we tackle this challenge by discovering smaller and more efficient models derived from a large base model. For additional details on Transformers, we refer the reader to the original Transformer paper (Vaswani et al., 2017) or the multiple surveys available on this topic (Lin et al., 2022).

Parameter-efficient fine-tuning (PEFT) techniques have been recently proposed to confront the challenges of fully fine-tuning large models, e.g., avoid having to update all the trainable weights of the pre-trained model or even the large number of parameters in a few selected layers (Ding et al., 2022). In addition to soft prompting (Lester et al., 2021) and prefix-tuning techniques (Li and Liang, 2021a), *adapters* have been proposed for PEFT in the past few years, initially in the form of *residual* adapters to allow convolutional neural networks to adapt to multiple visual domains (Rebuffi et al., 2017), and later for efficiently fine-tuning Transformer-based models (Houlsby et al., 2019; Stickland and Murray, 2019) to downstream NLP tasks. Using these adapters, we can freeze the original weights of the model and only update the parameters of the inserted adapters. There are several types of adapters. Sequential or serial adapters are placed between layers. These adapters have some drawbacks, including memory inefficiency. Parallel adapters address the limitations of sequential adapters (Pfeiffer et al., 2020). A variation of parallel adapters uses low-rank decomposition matrices for model adaptation, taking the name of Low-Rank Adapters (LoRA) (Hu et al., 2022). This paper integrates LoRA adapters into Neural Architecture Search, but the proposed framework could also integrate other PEFT adapters. Using LoRA adapters, a linear projection $\boldsymbol{Y}$, resulting from multiplying the input $\boldsymbol{X}$ and weights $\boldsymbol{W}$ (Equation 2), is extended by adding two low-rank adapters, $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ (Equation 3). The input is scaled by s. $\boldsymbol{L}_1$ is initialized with the standard Gaussian distribution, $\boldsymbol{L}_1 \sim \mathcal{N}(\mu, \sigma^2)$, with zero mean and unit variance, while $\boldsymbol{L}_2$ is initialized with all its entries equal to zero. The above process is formulated as follows:

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W}, \quad (2)$$

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{W} + s\boldsymbol{X}\boldsymbol{L}_1\boldsymbol{L}_2. \quad (3)$$

The main benefits of LoRA adapters during fine-tuning are obtained by freezing $\boldsymbol{W}$ and only updating the small number of parameters (compared to the number of the parameters of $\boldsymbol{W}$) in $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$, resulting in savings in memory consumption, for instance, because there is no need to compute gradients for the massive number trainable parameters in $\boldsymbol{W}$. Using adapters often results in trainable parameters that are fewer than 1% of the total parameters of the model. More recently, QLoRA (Dettmers et al., 2023) takes low-rank adapters a step further by proposing quantizing the frozen weights, resulting in additional savings in memory and storage.

Neural Architecture Search (NAS) research has increased significantly recently (Elsken et al., 2019; White et al., 2023). Given a set of possible architectures, NAS solutions attempt to find a high-performing architectural configuration that is more efficient than a baseline model. Initial proposals of NAS algorithms require partial or complete training of candidate architectures (Zoph and Le, 2017), which can be too costly. In the past few years, one-shot weight-sharing approaches have proven effective in discovering highly efficient architectures (Cai et al., 2020; Yu et al., 2020; Muñoz et al., 2022) that can be employed in complex deployment environments, e.g. (Yu et al., 2023). A benefit of the weight-sharing technique is that a super-network

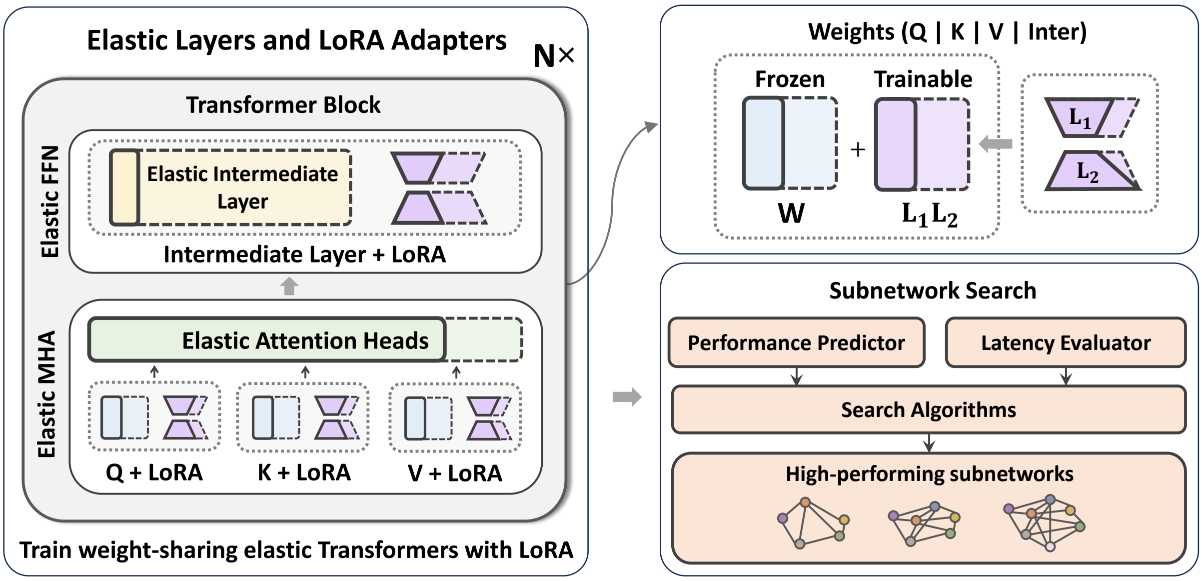


Figure 1: LoNAS' end-to-end workflow. Low-rank adapters are attached to Transformer blocks. LoNAS applies elasticity to the original model's frozen weights and the low-rank adapters.

composed of many subnetworks does not require additional memory and storage compared to the base model used to generate the super-network. In some cases, the minimal overhead relates to keeping track of the different values that can be used at each layer to activate alternative subnetworks.

An existing challenge that prevents NAS algorithms from being used with large models is that they often have to search for high-performing architectures in large design spaces, which would require a tremendous amount of resources in the case of large models. By incorporating PEFT, this paper demonstrates how weight-sharing NAS can discover smaller versions of relatively large language models with around 7 billion parameters. On a similar research path, AutoPEFT (Zhou et al., 2023) has proposed using Bayesian optimization on a search space of PEFT building blocks. Our approach, LoNAS, exploits the weight-sharing paradigm by enabling elasticity at the LoRA adapters and their dependent frozen weights. In the following sections, we describe LoNAS and present the results of generating super-networks for several language models.

3. Methodology

This section will delve into LoNAS, which integrates elastic LoRA adapters into NAS, resulting in high-performing compressed language models. As illustrated in Figure 1, LoNAS constructs a weight-sharing super-network by applying *elasticity* to a selection of layers (Section 3.1). The super-network is then optimized (Section 3.2) to improve the performance of its many subnetworks, and a subsequent search stage is conducted to discover high-performing subnetworks. The final subnetwork can be extracted resulting in a smaller model with fewer storage and memory requirements. LoNAS' end-to-end workflow is discussed in the following subsections.

3.1. Elasticity Alignment and Weight-Sharing Super-Network

The goal of LoNAS is to discover high-performing compressed models from larger models, which means that for some arbitrary linear layer, $\boldsymbol{l}_i$, LoNAS attempts to obtain smaller weight tensors, resulting in linear projections using a subset of the original parameters, in this case of the layer's weights, $\boldsymbol{W_i}$. For our purposes, *elasticity* means that the selected layer, $\boldsymbol{l}_i$, can have multiple values for a particular property, e.g., its width. For instance, if $\boldsymbol{W_i} \in \mathbb{R}^{m \times n}$, we might allow slicing this tensor to create subnetworks in which $\boldsymbol{l_i}$ activates k of its weights columns, s.t., $k < n$, resulting in an alternative version of the layer with a smaller width. As in other weight-sharing approaches, e.g., Once-for-all (Cai et al., 2020), BigNAS (Yu et al., 2020), LoNAS enables elasticity by masking the tensors of selected *elastic* layers, resulting in the activation of smaller subnetworks. However, different from these previous approaches, LoNAS efficiently integrates low-rank adapters into NAS, and since $\boldsymbol{W}$ remains frozen during fine-tuning, the NAS search space does not have to account for the possible configurations of $\boldsymbol{W}$ directly but only the possible configurations for the low-cost adapters. Formally, following Equation 3, we want to find slices $\boldsymbol{W}_\delta, \boldsymbol{L}_{\delta 1}, \boldsymbol{L}_{\delta 2}$ from $\boldsymbol{W}, \boldsymbol{L}_1$, and $\boldsymbol{L}_2$ such that,

$$\boldsymbol{W}_{\delta} \in \mathbb{R}^{h \times \{p_0, \ldots, p_m\}} \leftarrow \boldsymbol{W} \in \mathbb{R}^{h \times o}, \quad (4)$$

$$\boldsymbol{L}_{\delta 1} \in \mathbb{R}^{h \times \{s_0, \ldots, s_n\}} \leftarrow \boldsymbol{L}_1 \in \mathbb{R}^{h \times r}, \quad (5)$$

$$\boldsymbol{L}_{\delta 2} \in \mathbb{R}^{s \times \{p_0, \ldots, p_m\}} \leftarrow \boldsymbol{L}_2 \in \mathbb{R}^{r \times o}, \quad (6)$$

where $p_i \leq o$, $s_i \leq r$ and $r \ll o$. There might be several possible values for p_i and s_i, resulting in a rich search space of subnetwork configurations. Memory consumption is reduced due to $\boldsymbol{W}$ remaining frozen. The only trainable parameters explored by LoNAS are the subsets of $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$. However, this requires the management of dependencies between the adapters' elasticity and the corresponding weights' elasticity, which is discussed next.

Dependency Groups When activating a subnetwork, the subset $\boldsymbol{W}_{\delta} \subseteq \boldsymbol{W}$ (Equation 4) is strictly dependent on the subset $\boldsymbol{L}_{\delta 2} \subseteq \boldsymbol{L}_2$. When choosing an elastic configuration, LoNAS maintains these structures with consistent shapes. $\boldsymbol{L}_1$, on the other hand, can be sliced arbitrarily without aligning with the original model's frozen weights. In the case of the multi-attention heads, LoNAS enables elasticity and the possibility of attaching adapters to the **Q**, **K**, and **V** layers. We explore several configurations in Section 4.

As illustrated in Figure 1, LoNAS also enables elastic adapters in layers of the Multilayer Perceptron (MLP) that follow each multi-head attention block. LoNAS enables elasticity directly on intermediate layers of the MLP or in the adapters attached to these layers. In both cases, the goal is to obtain subnetworks with subsets of weights of the larger model. These efficient subnetworks have a reduced footprint compared to the original model.

3.2. Fine-Tuning the Weight-Sharing Super-Network

Once elasticity has been enabled, resulting in a fixed search space of possible configurations for $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ at each of the selected layers, we need to pay particular attention to super-network training. We sample random subnetworks for each data batch at each iteration, as Yu et al. (2020) recommended. We must consider the frozen weights, $\boldsymbol{W}$, during the forward pass. However, during the backward pass, we only need to compute the gradients of the elastic adapters to update their parameters in the super-network.

4. Evaluation

In this section, we conduct experiments by generating super-networks for multiple language models and testing some of their subnetworks on several datasets to evaluate the effectiveness of LoNAS. Initially, we investigate the application of LoNAS on a small-scale language model employed in the early phases of our research. Subsequently, we continue our exploration of LoNAS on larger language models. The details of our setup and the analysis of the results are discussed next.

4.1. Experimental Setup

4.1.1. Datasets

Regarding our experimentation with a small language model, we utilized the General Language Understanding Evaluation (GLUE) benchmark (Wang et al., 2019) to initiate our exploration of LoNAS. The GLUE benchmark is a widely-used evaluation framework in the field of NLP, including eight tasks RTE (Dagan et al., 2005), MRPC (Dolan and Brockett, 2005b), STS-B (Cer et al., 2017), CoLA (Warstadt et al., 2018), SST-2 (Socher et al., 2013), QNLI (Wang et al., 2019), QQP (Chen et al., 2017), and MNLI (Williams et al., 2018).

In the context of our experimentation with large language models, we compare our LoNAS results with those reported in the LLM-Adapters paper (Hu et al., 2023) for 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), as well as four math reasoning datasets: GSM8K (Cobbe et al., 2021), AQUA (Ling et al., 2017), MAWPS (Lan et al., 2022), SVAMP (Patel et al., 2021). In our experiments, we utilized the training data provided by the LLM-Adapters group, which combines multiple training datasets into a unified dataset for training one general model and subsequently tests across each dataset. Moreover, the unified datasets they generated are extracted with the help of Zero-shot Chain of Thought (CoT) (Wei et al., 2022) and GPT-3 text-davinci003[1].

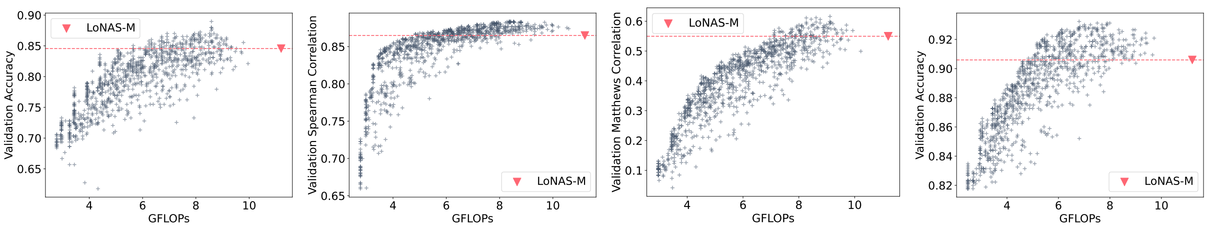
4.1.2. Models

In our initial experiments, we assess LoNAS employing a small language model, the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019). We present these findings in this section to demonstrate the impact of LoNAS on smaller models, which are usually employed to demonstrate other NAS and PEFT frameworks, e.g., AutoPEFT (Zhou et al., 2023). Regarding the application of larger language models, we generate LoNAS super-networks for the representative open-source autoregressive text generation LLMs LLaMA$_{7B}$ and BLOOMz$_{7.1B}$, boasting a total of 6.7 and 7.1 billion parameters, respectively. LLaMA was trained by Meta using data in 20

[1]https://platform.openai.com/docs/models/gpt-3-5

Table 1: Results on GLUE tasks with $\text{BERT}_{\text{base}}$. We report the best development set performance. Green and blue represent the best and second-best scores, respectively. The baseline results are sourced from Zhou et al. (2023). Consistent with previous work, we present Spearman's correlation for STS-B, Matthew's correlation coefficient for CoLA, and accuracy metrics for the remaining tasks. Additionally, we provide the average GFLOPs of the discovered sub-network for each task.

Method	**Parameter Ratios**	**GFLOPs**	**GLUE Benchmark**								**AVG**
			RTE	**MRPC**	**STS-B**	**CoLA**	**SST-2**	**QNLI**	**QQP**	**MNLI**	
FFT	100%	11.2	71.12	85.74	89.00	59.32	92.57	91.50	91.52	84.43	83.15
Prefix	0.17%	11.2	70.54	85.93	88.76	58.88	91.93	90.76	89.12	82.78	82.33
LoRA	0.27%	11.2	65.85	84.46	88.73	57.58	92.06	90.62	89.41	83.00	81.46
Serial	0.81%	11.2	68.01	84.75	88.61	59.73	91.93	91.06	90.52	84.18	82.35
AdaMix	0.81%	11.2	70.11	86.86	89.12	59.11	92.06	91.52	90.22	84.25	82.91
UniPELT	1.25%	11.2	67.07	84.22	88.84	60.13	92.52	91.09	90.69	84.28	82.35
Parallel	6.46%	11.2	68.52	86.52	88.90	58.72	92.13	90.83	90.74	73.93	81.29
MAM	6.97%	11.2	69.10	87.16	89.01	47.87	83.94	90.85	90.76	83.31	80.25
AutoPEFT	1.40%	11.2	72.35	87.45	89.17	60.92	92.22	91.12	90.64	84.01	**83.49**
LoNAS	0.27%	8.0	70.76	88.97	88.28	61.12	93.23	91.21	88.91	82.00	**83.06**



(a) MRPC (b) STS-B (c) CoLA (d) SST-2

Figure 2: Search progression on the super-network trained with $\text{BERT}_{\text{base}}$ + LoNAS on four tasks of the GLUE benchmark. Subnetworks are sampled from the super-network by the Non-Dominated Sorting Genetic Algorithm (NSGA-II) (Deb et al., 2002), resulting in a Pareto front of efficient configurations.

languages, but most of the text is in English. The data used for training LLaMA primarily originates from the CCNet (Wenzek et al., 2019) and C4 (Raffel et al., 2020b) datasets making up ≈82% of the training data. The remaining data includes other sources like GitHub and Wikipedia. LLaMA_{7B} has 32 layers and 32 heads in each multi-head attention layer. $\text{BLOOMz}_{7.1B}$ was trained by Scao et al. (2022) using the ROOTS corpus (Laurençon et al., 2022) across 59 languages, featuring 30 Transformer blocks and 32 heads in each multi-head attention layer. Further details are available on the Hugging Face model cards [2],[3].

4.1.3. Baselines

We compare LoNAS subnetworks with other models using several representative types of adapters: (1) Prefix-tuning (Li and Liang, 2021b) integrates soft prompts into the hidden states across all layers. (2) Series adapter (Houlsby et al., 2019) integrates additional learnable modules sequentially within a specific sublayer. (3) Parallel adapter (Pfeiffer et al., 2020) is placed at the level of the MHA or MLP layers. (4) LoRA (Hu et al., 2022) is the low-rank parallel adapter placed at the same level of the linear layers of the Transformer block while keeping the original weights $\boldsymbol{W}$ of the Transformer block frozen. To compare LoNAS to other complex PEFT adapters, we incorporate the results of more complex frameworks, including AdaMix (Wang et al., 2022), UniPELT (Mao et al., 2021), MAM (Liao et al., 2023), AutoPEFT (Zhou et al., 2023), and full fine-tuning (FFT).

4.1.4. Implementation Details

LoNAS is implemented utilizing OpenVINO's Neural Network Compression Framework[4] and its BootstrapNAS solution (Muñoz et al., 2022). We modified this library to enable hooks in the frozen

[2]https://huggingface.co/yahma/llama-7b-hf
[3]https://huggingface.co/bigscience/bloomz-7b1

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

Table 2: Results on eight commonsense reasoning tasks with LLaMA_{7B}. We compare LoNAS' efficiency and test accuracy(%) with other LLM-Adapter approaches using the 15k unified commonsense reasoning data (Hu et al., 2023) for training. LoNAS-M represents the maximal subnetwork and LoNAS-H is a subnetwork obtained with the midpoint heuristic (Equation 7).

Method	Total Params.	TFLOPs	Commonsense Reasoning - Accuracy (%)								Average
			BoolQ	PIQA	SIQA	HellaSwag	WinoG	Arc-e	Arc-c	OBQA	
ChatGPT	-	-	73.1	85.4	68.5	78.5	66.1	89.8	79.9	74.8	77.0
Prefix	6.7B	1.7	55.2	57.5	49.4	34.5	36.2	60.0	47.8	41.2	47.7
Series	6.7B	1.7	62.2	71.8	66.3	39.2	58.5	73.3	54.4	67.8	61.7
Parallel	6.7B	1.7	62.2	72.6	66.3	47.9	57.4	75.1	56.2	67.6	63.2
LoRA	6.7B	1.7	62.6	75.3	67.9	52.9	58.6	79.2	58.3	71.2	**65.8**
LoNAS-M	6.7B	1.7	65.8	77.3	72.2	57.0	67.6	79.0	61.9	77.4	**69.8**
LoNAS-H	5.6B	1.4	62.9	73.0	68.7	51.4	63.9	72.3	58.5	71.0	65.2

weights and their alignment with the corresponding inserted low-rank adapters in LoNAS. We also patch Hugging Face's PEFT library[5] to enable elasticity at the low-rank adapters. More details about the hyperparameters and experiments with other search spaces are included in the supplementary material.

4.2. Initial Experimentation on a Small Language Model

To explore the benefits of LoNAS, we initially experimented with a small language model, BERT. Table 1 compares a discovered LoNAS subnetwork, full fine-tuning (FFT), and other PEFT adapters on the GLUE benchmark. As LoRA adapters, LoNAS yields subnetworks with fewer trainable parameters, i.e., only 0.27% of the total number of parameters. However, among all evaluated adapters, only the Prefix adapter approach exhibits a lower count of trainable parameters (0.17% of the total number of parameters). Among the methods with fewer trainable parameters (Prefix, LoRA, and LoNAS), our approach achieves smaller and more efficient models compared to LoRA and prefix (GFLOPs 8.0 vs. 11.2), with higher average (83.06 vs. 81.46 and 82.33). Compared to other more advanced PEFT methods, LoNAS distinguishes itself for its fewer trainable parameters, more efficient models, and the ability to maintain comparable accuracy performance. Moreover, we present some visualization results depicting the progression of the search in Figure 2, illustrating the presence of optimal subnetworks within the trained super-network that are smaller and more efficient and deliver accuracy levels surpassing those of larger subnetworks.

[5]https://github.com/huggingface/peft.git

4.3. Performance on Large Language Models

Based on the previous BERT experimental results, we can observe the benefits of LoNAS on small-scale models. In this section, we will explore the performance of LoNAS on larger models. In general, once a super-network has been trained, we could utilize the Non-Dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al., 2002) or any other alternative search algorithm to discover a Pareto frontier of high-performing subnetworks, and finally a subset of subnetworks that enhance both accuracy and efficiency while satisfying user-defined requirements can be selected. However, employing NSGA-II becomes expensive when dealing with larger models. To quickly estimate the quality of the smaller subnetworks, we employ a heuristic approach (Muñoz et al., 2024a) in which from all the n possible configurations of an elastic layer l_i, this method selects a configuration indexed at c, close to the midpoint of the range of possible configurations, formulated as follows:

$$\textbf{LoNAS-H}_{l_i} \leftarrow \textbf{LoNAS-M}_{l_i}[c], \text{s.t. } c = \left\lfloor \frac{n}{2} \right\rfloor. \quad (7)$$

LoNAS-M is the configuration of the maximal subnetwork, which is equivalent in its architecture to the original base model. LoNAS-H is the subnetwork configuration obtained by the heuristic. Section 4.4 compares this heuristic to the evolutionary search and discusses the trade-offs in selecting each approach.

4.3.1. Commonsense Reasoning with LLaMA

As Table 2 shows, LLaMA_{7B} + LoNAS-M outperforms all baselines in all datasets, while LLaMA_{7B} + LoNAS-H uses fewer parameters and TFLOPs with a minor drop in accuracy compared to LoRA. Based on the initial results using the heuristic, the find-

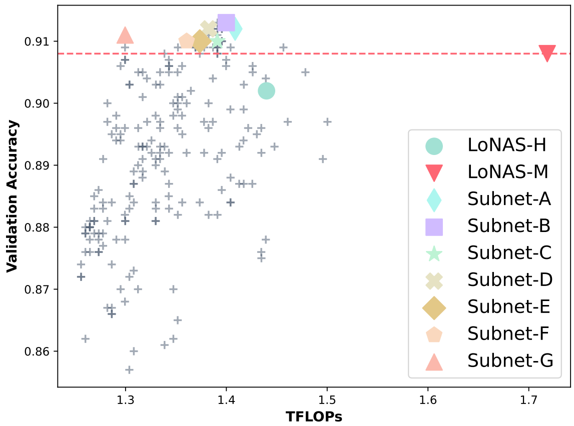


Figure 3: Search progression on the super-network trained with $LLaMA_{7B}$ + LoNAS on the unified commonsense reasoning dataset. Additional high-performing subnetworks are available from the Pareto frontier based on the desired performance in the accuracy/TFLOPs multi-objective space.

ings of LLaMA on commonsense reasoning demonstrate the benefits of LoNAS on large-scale models, indicating its capability to achieve more efficient models while maintaining accuracy. Section 4.4 presents a comprehensive analysis of the evolutionary search conducted with the $LLaMA_{7B}$ + LoNAS super-network in this setting. Further search results illustrate the discovery of better subnetworks within the super-network facilitated by advanced search algorithms. Notably, these subnetworks exhibit enhanced efficiency with superior performance metrics.

4.3.2. Math Reasoning with BLOOMz

To further explore the performance of LoNAS on other LLMs and tasks, we extended our investigation to another prominent open-source LLM, BLOOMz, and delved into a distinct downstream task within the field of math. The performance metrics of $BLOOMz_{7B}$ + LoNAS across four math reasoning tasks are presented in Table 3. Notably, the results demonstrate that LoNAS can produce the model (LoNAS-H) of heightened efficiency while maintaining competitive accuracy within this experimental setting. It is important to emphasize that our experimentation encompassed a broader range of evaluations on the math dataset, revealing instances where LoNAS' performance does not perform well in other LLMs, exposing inherent limitations within our approach. More details can be found in Appendix D.

4.4. Analysis of Subnetwork Evolutionary Search

To better illustrate the subnetwork search stage, Figure 3 shows the search progression with NSGA-II in which the performance of sampled subnetwork configurations from the $LLaMA_{7B}$ super-network are plotted in a multi-objective space of accuracy and TFLOPs. As described in section 4.1, we devise a midpoint heuristic (Equation 7) to quickly explore the quality of our fine-tuned super-networks and discover a smaller subnetwork at an approximately central region of the search space. However, considering accuracy and efficiency, the subnetwork obtained with this approach is not likely the best. LoNAS can discover better subnetworks from the one discovered using the midpoint heuristic by applying the more advanced evolutionary search using the NSGA-II algorithm to explore our trained super-network further. This figure illustrates the search progression on the $LLaMA_{7B}$ + LoNAS super-network trained with the unified commonsense reasoning dataset. We use the accuracy on the validation set as a metric to guide the search process. After conducting an evolutionary search, we select a few subnetworks with reasonable accuracy on the validation set to further investigate their performance on the test dataset. As shown in Table 4, we can obtain smaller, more efficient models with higher test accuracy than $LLaMA_{7B}$ + LoNAS-H and $LLaMA_{7B}$ + LoRA, e.g., Subnet-B and Subnet-D.

We notice an interesting observation: the performance on the validation set does not consistently correlate with that on the test set, which implies the validation set might not provide accurate guidance in identifying the optimal subnetwork for performance on the test set. We attribute this divergence to two factors. Firstly, the validation set is randomly sampled from the unified dataset comprising eight individual commonsense reasoning datasets, leading to inherent randomness in the distribution of datasets within the validation set. Employing a validation set tailored to each specific dataset could yield improved results, and we plan to investigate this hypothesis in future research. Secondly, the generalization capacity of a subnetwork should also be related to its size rather than solely considering performance metrics derived from a limited number of validation samples. To strike a balance, it is important to weigh the validation set accuracy against efficiency metrics, such as FLOPs, to align with our objectives. Notably, in the LoNAS method, users do not need to retrain different models when facing diverse demands. Instead, extracting subnetworks from a once-trained super-network that meets various requirements suffices.

Table 3: Results on four math reasoning tasks with $\text{BLOOMz}_{7.1B}$. The results from Prefix, Series, Parallel, and LoRA baseline are those reported by Hu et al. (2023). LoNAS-M denotes the maximal subnetwork, while LoNAS-H refers to a subnetwork derived using the midpoint heuristic approach.

Method	Total Params.	TFLOPs	Math Reasoning - Accuracy (%)				Average
			GSM8K	AQuA	MAWPS	SVAMP	
Prefix	7.1B	1.8	13.8	12.5	47.5	24.1	24.5
Series	7.1B	1.8	14.3	20.5	62.2	38.1	33.8
Parallel	7.1B	1.8	18.5	18.9	70.6	36.4	36.1
LoRA	7.1B	1.8	17.4	21.3	70.2	41.0	**37.5**
LoNAS-M	7.1B	1.8	20.8	6.3	76.9	32.2	34.0
LoNAS-H	6.1B	1.5	18.6	22.0	76.5	31.8	**37.2**

Table 4: Efficiency and accuracy comparison of multiple selected LLaMA_{7B} + LoNAS subnetworks, corresponding to the points shown in Figure 3. The validation set is randomly sampled from the unified training data set, which is used to guide the search process. The test accuracy is the average accuracy on eight commonsense reasoning datasets. Inference speedup is relative to the maximal subnetwork with the same configuration as the base model LLaMA_{7B}. We use an Intel Xeon Platinum 8480L with Intel Extension for PyTorch (IPEX) enabled to collect the average token latency (100 generation iterations).

Search Method	Subnetwork	TFLOPs	Validation Accuracy(%)	Test Accuracy(%)	Inference Speedup		
					FP32	BF16	INT8
Maximal	LoNAS-M	1.72	90.8	69.8	1.00×	2.84×	4.20×
Heuristic	LoNAS-H	1.44	90.2	65.2	1.23×	3.14×	4.74×
Evolutionary (NSGA-II)	Subnet-A	1.41	91.2	67.1	1.26×	3.17×	4.81×
	Subnet-B	1.40	91.3	67.1	1.28×	3.18×	4.83×
	Subnet-C	1.39	91.0	66.9	1.29×	3.19×	4.85×
	Subnet-D	1.38	91.2	67.0	1.30×	3.21×	4.88×
	Subnet-E	1.37	91.0	66.6	1.31×	3.22×	4.90×
	Subnet-F	1.36	91.0	65.9	1.32×	3.23×	4.92×
	Subnet-G	1.29	91.1	65.6	1.41×	3.32×	5.09×

Cost of Evolutionary Search Although we obtained a more efficient model with good validation accuracy in the unified dataset, running an evolutionary search for LLMs is significantly costlier. In our experiments, the number of samples in the validation set is 1000, and the number of subnetworks evaluated is 200 (i.e., Except for LoNAS-M and LoNAS-H, there are 200 points in Figure 3.). Each evaluation for one LLaMA_{7B} + LoNAS subnetwork took approximately 10 minutes, and completing the search stage using a single GPU typically requires 1 to 2 days. However, the cost of the search progression can be amortized by all the savings related to having a smaller model during the inference stage. The user should decide between the approach using the proposed heuristic or an expensive evolutionary search based on the requirements and optimization budgets and consider the trade-off between search cost and subnetwork quality.

4.5. Inference Benefit Analysis of LoNAS

LoRA provides benefits during training/fine-tuning since the weights $\boldsymbol{W}$ of the large model are frozen, and the number of trainable parameters and memory requirements are reduced. LoNAS takes these benefits further during inference by obtaining compressed models with lower latency than the original model while having similar or even better accuracy in the validation and test datasets. As described in Table 4, the subnetwork obtained using our heuristic LoNAS-H and the subnetworks obtained after evolutionary search have fewer total parameters than the base model, which has the same architecture as the subnetwork LoNAS-M. The compressed subnetworks result in a speedup during inference of up to 1.41, 3.32, and 5.09 times the inference time of the base model for FP32, BF16, and INT8 precision types, respectively. We enable Intel Extension for PyTorch (IPEX)[6] to accelerate inference in CPU.

[6]https://github.com/intel/intel-extension-for-pytorch

The results show that further speedup is obtained by quantizing the discovered subnetworks.

5. Discussion and Future Work

More Efficient Applications of Elasticity in Adapters We have observed that LoNAS fails to generalize to other architectures, e.g., GPT-J (Wang and Komatsuzaki, 2021) and datasets, e.g., unified math reasoning (Hu et al., 2023), and requires costly training and subnetwork discovery stages. As a follow-up to this work, we are further improving LoNAS by applying elasticity only at the adapter level and leaving the original weights of the model untouched (Muñoz et al., 2024a). Although this variation of LoNAS does not obtain a significant reduction in the model size, it preserves or improves the accuracy.

Quantization of the Frozen Weights Quantization can also benefit LoNAS' training efficiency. Recently, QLoRA (Dettmers et al., 2023) proposed an extension to the classical LoRA formulation (Hu et al., 2022), in which the frozen weights, $\boldsymbol{W}$, are quantized to 4-bit NormalFloat (NF4) precision toreduce memory consumption further (Equation 8). QLoRA also quantizes the quantization constants, c_i, further reducing the memory footprint during training. QLoRA can be formulated as follows:

$$\boldsymbol{Y}^{\alpha} = \boldsymbol{X}^{\alpha} D(c_1^{\beta}, c_2^{k\text{-}bit}, \boldsymbol{W}^{\gamma}) + \boldsymbol{X}^{\alpha} \boldsymbol{L}_1^{\alpha} \boldsymbol{L}_2^{\alpha} \tag{8}$$

$$D(c_1^{\beta}, c_2^{k\text{-}bit}, \boldsymbol{W}^{\gamma}) = d(d(c_1^{\beta}, c_2^{k\text{-}bit}), \boldsymbol{W}^{\gamma}) = \boldsymbol{W}^{\alpha}, \tag{9}$$

where α, β, and γ indicate BF16 precision, FP32 precision, and NF4 precision, respectively. D is the Double Dequantize operation that first dequantizes (d in Equation 9) constants c_1, and c_2 and then dequantizes $\boldsymbol{W}^{\gamma}$. Quantization of the frozen weights is out of the scope of this paper. However, our future plans are to improve LoNAS' efficiency further. Quantization and other techniques will help LoNAS continue reducing its footprint during training, QLoRA has done.

Weight reordering Weight reordering strategies are often used in weight-sharing NAS to improve the quality of the super-networks (Muñoz et al., 2024b). This step can be costly when working with LLMs. We are interested in investigating efficient approaches to weight reordering for large models.

6. Conclusion

This paper demonstrates a novel approach, LoNAS, to integrate Parameter-Efficient Fine-Tuning (PEFT) techniques and low-rank (LoRA) adapters, in particular, into Neural Architecture Search (NAS) for LLMs. LoNAS enables elasticity in low-rank adapters and their corresponding frozen weights in the base model. The generated super-network is then efficiently fine-tuned with fewer than 1% of the total parameters. A subsequent search stage discovers smaller compressed subnetworks that reduce the resource requirements for deploying these models. Hence, LoNAS demonstrates that we can increase the deployment range of the original larger models. LoNAS subnetworks have several implications for improving the inference stage, e.g., less memory consumption and speedup when using smaller models. As indicated above in the future work discussion, there are many research paths we plan to explore to improve further the efficiency of the discovered subnetworks during inference. In the following section, we include a discussion of the limitations that we have observed in LoNAS. The code is available at https://github.com/IntelLabs/Hardware-Aware-Automated-Machine-Learning.

Limitations

LoNAS is an initial exploration of the combination of NAS and PEFT. We have observed that the approach struggles to generalize to other models and datasets. For instance, LoNAS-LLaMA struggles to obtain good results in the math reasoning dataset, while LoNAS-BLOOMz struggles in the commonsense dataset. To address these and other issues, as a follow-up of this work, we propose a modification of LoNAS that applies NAS only to the adapters and that benefits from a pre-fine-tuning stage in which the given model is sparsified (Muñoz et al., 2024a). Another limitation that should be explored further in future work is the high cost of searching for the subnetworks when using approaches such as evolutionary search. Although this high cost can be amortized with the gains in a more efficient inference stage due to the discovered smaller models (Section 4.4), it requires more investigation, mainly when working with LLMs.

Ethics Statement

Large language models (LLMs) are relatively new technologies with challenges and limitations. Despite all the success that LLMs have had and their integration into popular applications, we must be aware of the risks and harm that LLMs might bring upon some people, e.g., by producing inaccurate responses and misinformation that could negatively impact them. These limitations are outside the scope of this paper. LoNAS aims to deliver compressed models that can run efficiently on resource-constrained devices. However, the limitations mentioned above (and others omitted here) must be taken into account when designing real-world systems and applications that use large language models. Before deploying these models, it is imperative to conduct exhaustive testing and identify risks and vulnerabilities to prevent any potential harm.

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 the process of open-sourcing our compressed models. We also thank the anonymous reviewers for their insightful suggestions, which helped us improve the paper.

7. Bibliographical References

Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. 2021. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. 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 7319–7328, Online. Association for Computational Linguistics.

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*.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, S. Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen A. Creel, Jared Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas F. Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, O. Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir P. Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Benjamin Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, J. F. Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Robert Reich, Hongyu Ren, Frieda Rong, Yusuf H. Roohani, Camilo Ruiz, Jack Ryan, Christopher R'e, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishna Parasuram Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei A. Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models. *ArXiv*.

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*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 1–14, Vancouver, Canada. Association for Computational Linguistics.

Zihang Chen, Hongbo Zhang, Xiaoji Zhang, and Leqi Zhao. 2017. Quora question pairs.

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 *The North American Chapter of the Association for Computational Linguistics (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.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Machine Learning Challenges Workshop*, pages 177–190. Springer.

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.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

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.

Bill Dolan and Chris Brockett. 2005a. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*. Asia Federation of Natural Language Processing.

William B. Dolan and Chris Brockett. 2005b. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

William B. Dolan and Chris Brockett. 2005c. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

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*.

Denis Kocetkov, Raymond Li, Loubna Ben allal, Jia LI, Chenghao Mou, Yacine Jernite, Margaret Mitchell, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro Von Werra, and Harm de Vries. 2023. The stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Romero Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Vu Minh Chien, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Luccioni, and Yacine Jernite. 2022. The bigscience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021a. 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)*. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021b. 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.

Baohao Liao, Shaomu Tan, and Christof Monz. 2023. Make your pre-trained model reversible: From parameter to memory efficient fine-tuning.

Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2022. A survey of transformers. *AI Open*, 3:111–132.

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. *arXiv preprint arXiv:2305.11627*.

Yuning Mao, Lambert Mathias, Rui Hou, Amjad Almahairi, Hao Ma, Jiawei Han, Wen tau Yih, and Madian Khabsa. 2021. Unipelt: A unified framework for parameter-efficient language model tuning.

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

J. Pablo Muñoz, Jinjie Yuan, and Nilesh Jain. 2024a. Shears: Unstructured sparsity with neural low-rank adapter search. Accessed: 2024-03-05.

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.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2022. Efficiently scaling transformer inference.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

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

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. 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.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martínez Alonso. 2014. What's in a p-value in NLP? In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*, pages 1–10, Ann Arbor, Michigan. Association for Computational Linguistics.

Asa Cooper Stickland and Iain Murray. 2019. BERT and PALs: Projected attention layers for efficient adaptation in multi-task learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5986–5995. PMLR.

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.

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.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In the Proceedings of ICLR.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Yaqing Wang, Sahaj Agarwal, Subhabrata Mukherjee, Xiaodong Liu, Jing Gao, Ahmed Hassan Awadallah, and Jianfeng Gao. 2022. Adamix: Mixture-of-adaptations for parameter-efficient model tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. 2019. Ccnet: Extracting high quality monolingual datasets from web crawl data. *arXiv preprint arXiv:1911.00359*.

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.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. 2022. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *AAAI*.

Jiahui Yu and Thomas S. Huang. 2019. Universally slimmable networks and improved training techniques. *CoRR*, abs/1903.05134.

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.

Sixing Yu, J Pablo Muñoz, and Ali Jannesari. 2023. Federated foundation models: Privacy-preserving and collaborative learning for large models. *arXiv preprint arXiv:2305.11414*.

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*.

Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. 2023. Lora-fa: Memory-efficient low-rank adaptation for large language models fine-tuning.

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

Barret Zoph and Quoc Le. 2017. Neural architecture search with reinforcement learning. In *International Conference on Learning Representations*.

A. Hyperparameters

The hyperparameters of our experiments for several language models are listed in Table 5 and Table 6.

B. Search Spaces

The search spaces for the super-networks across different language models is shown in Table 7 and Table 8.

C. More Latency Comparison

More latency comparisons of different precisions, first token generation and second token generation are shown in Table 9.

D. Other Experiments with the Unified Math Dataset

From the main paper, we have observed some benefits of the LoNAS approach. However, in our exploration of LoNAS, we have also encountered limitations. Specifically, we have investigated the performance of the LLaMA-series model + LoNAS on math datasets. As illustrated in Table 10, it can be observed that the combination of LLaMA-series models with LoNAS does not exhibit good performance in the math domain - as the scale of the subnetwork models decreases, the performance of the models also decreases. This observation exposes some drawbacks and limitations of LoNAS, indicating a lack of generalizability. As stated in the LLM-pruner (Ma et al., 2023) paper, compressing large language models under high compression rates remains a significant challenge. This paper initiates an exploration of PEFT + NAS on LLMs, hoping to provide some insights to the research community.

Table 5: Hyperparameters for the experiments with $BERT_{base}$. For all experiments, we use LoRA with a value of 8 for low rank and 16 for alpha. The target modules for LoRA are query and value.

Task	**RTE**	**MRPC**	**STS-B**	**CoLA**	**SST-2**	**QNLI**	**QQP**	**MNLI**
Epoch	80	35	60	80	60	80	60	40
Batch size	32	32	64	64	64	64	64	64
Learning rate	3e-4	5e-4	5e-4	3e-4	3e-4	4e-4	3e-4	4e-4
Max length	128	128	128	128	128	256	128	128

Table 6: Hyperparameters for the experiments with $LLaMA_{7B}$ and $BLOOMz_{7B}$.

Model	**$LLaMA_{7B}$**	**$BLOOMz_{7B}$**
Epoch	6	8
Batch size	16	16
Learning rate	3e-4	3e-4
LoRA r	32	32
LoRA alpha	64	64
LoRA target modules	q_proj, k_proj, v_proj, up_proj, gate_proj, down_proj	query_key_value, dense_h_to_4h, dense_4h_to_h

Table 7: Search spaces for the $BERT_{base}$ super-network.

Layer	**Q & K & V & Q-LoRA$_{L2}$ & V-LoRA$_{L2}$**	**Q-LoRA$_{L1}$ & V-LoRA$_{L1}$**	**Intermediate Dense**
0	[768, 384]	[8, 4, 2]	[3072, 2634, 216]
1	[768, 320]	[8, 4, 2]	[3072, 2634, 181]
2	[768, 256]	[8, 4, 2]	[3072, 2627, 208]
3	[768, 512]	[8, 4, 2]	[3072, 2676, 226]
4	[768, 512]	[8, 4, 2]	[3072, 2628, 179]
5	[768, 704]	[8, 4, 2]	[3072, 2662, 175]
6	[768, 576]	[8, 4, 2]	[3072, 2706, 182]
7	[768, 576]	[8, 4, 2]	[3072, 2687, 169]
8	[768, 640]	[8, 4, 2]	[3072, 2616, 165]
9	[768, 192]	[8, 4, 2]	[3072, 2400, 160]
10	[768, 704, 192]	[8, 4, 2]	[3072, 2198, 163]
11	[768, 320]	[8, 4, 2]	[3072, 1940, 150]

Table 8: Search spaces for the $LLaMA_{7B}$ and $BLOOMz_{7B}$ super-networks. All layers share the same search space.

Model	**Q-LoRA$_{L1}$ & K-LoRA$_{L1}$ & V-LoRA$_{L1}$**	**Up & Gate & Up-LoRA$_{L2}$ & Gate-LoRA$_{L2}$**	**Up-LoRA$_{L1}$ & Gate-LoRA$_{L1}$**
$LLaMA_{7B}$	[32, 28]	[11008, 9632, 8256, 6880, 5504]	[32, 28]
Model	**QKV-LoRA$_{L1}$**	**Dense_h_to_4h & Dense_h_to_4h-LoRA$_{L2}$**	**Dense_h_to_4h-LoRA$_{L1}$**
$BLOOMz_{7B}$	[32, 28]	[16384, 14336, 12288, 10240, 8192]	[32, 28]

Table 9: Further latency comparison beyond Table 4. We test the inference speedup on FP32, BF16, INT8 and INT4, and also test the latency for first token generation. We use an Intel Xeon Platinum 8480L with Intel Extension for PyTorch (IPEX) enabled to collect the avarage latency (100 generation iterations).

Search Method	Subnetwork	TFLOPs	First Token Latency				Second Token Latency			
			FP32	BF16	INT8	INT4	FP32	BF16	INT8	INT4
Maximal	LoNAS-M	1.72	1.00×	3.41×	3.63×	5.04×	1.00×	2.84×	4.20×	5.29×
Heuristic	LoNAS-H	1.44	1.11×	3.87×	4.38×	5.27×	1.23×	3.14×	4.74×	5.97×
Evolutionary (NSGA-II)	Subnet-A	1.41	1.13×	3.92×	4.48×	5.30×	1.26×	3.17×	4.81×	6.05×
	Subnet-B	1.40	1.13×	3.94×	4.51×	5.31×	1.28×	3.18×	4.83×	6.08×
	Subnet-C	1.39	1.14×	3.96×	4.55×	5.32×	1.29×	3.19×	4.85×	6.11×
	Subnet-D	1.38	1.14×	3.98×	4.58×	5.32×	1.30×	3.21×	4.88×	6.14×
	Subnet-E	1.37	1.15×	4.00×	4.62×	5.33×	1.31×	3.22×	4.90×	6.17×
	Subnet-F	1.36	1.15×	4.02×	4.65×	5.34×	1.32×	3.23×	4.92×	6.20×
	Subnet-G	1.29	1.19×	4.16×	4.92×	5.40×	1.41×	3.32×	5.09×	6.41×

Table 10: Results on four math reasoning tasks with LLaMA_{7B} and LLaMA_{13B}. The results from the Prefix, Series, Parallel, and LoRA baselines are those reported by Hu et al. (2023). LoNAS-M denotes the maximal subnetwork, while LoNAS-H refers to a subnetwork derived using a heuristic approach.

LLM	Method	TFLOPs	Total Params.	Math Reasoning - 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	1.7	6.7B	24.4	14.2	63.4	38.1	35.0
	Series	1.7	6.7B	33.3	15.0	77.7	**52.3**	44.6
	Parallel	1.7	6.7B	35.3	18.1	**82.4**	49.6	46.4
	LoRA	1.7	6.7B	**37.5**	18.9	79.0	52.1	**46.9**
	LoNAS-M	1.7	6.7B	36.7	19.7	81.9	47.8	46.5
	LoNAS-H	**1.4**	**5.6B**	30.2	**20.5**	81.1	43.2	43.7
LLaMA_{13B}	Prefix	3.3	12.9B	31.1	15.7	66.8	41.4	38.8
	Series	3.3	12.9B	44.0	**22.0**	78.6	50.8	48.9
	Parallel	3.3	12.9B	43.3	20.5	81.1	**55.7**	50.2
	LoRA	3.3	12.9B	**47.5**	18.5	**83.6**	54.6	**51.1**
	LoNAS-M	3.3	12.9B	46.9	20.9	83.2	53.0	51.0
	LoNAS-H	**2.8**	**10.8B**	40.0	19.7	82.8	51.9	48.6