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

Anonymous authors
Paper under double-blind review

Abstract

Large Language Models (LLMs) continue to grow, reaching hundreds of billions of parameters and making it difficult for Deep Learning practitioners with resource-constrained systems to use them, e.g., fine-tune these models for a downstream task of their interest. Adapters, e.g., 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 train or 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 large 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, LoNAS models contain fewer total parameters, reducing the inference time while improving accuracy. LoNAS subnetworks are available to reviewers to reproduce our results at: **<anonymous Git repository>**

Keywords: NLP, Transformers, Parameter-Efficient Fine-Tuning, Neural Architecture Search

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 smaller versions of large models 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, model size is still a problem for many NLP practitioners.

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 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 the deployment at 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 NAS 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.

2. LoNAS efficiently compresses large models, resulting in Pareto frontiers of smaller model variants, which save memory and computation, allowing for faster inference and increasing the range of devices to deploy these models.

3. Extensive experiments to study the performance of LoNAS on six reasoning datasets. LoNAS outperforms all baseline methods in both efficiency and accuracy. Our code will be open-sourced for further experimentation by the research community.

2. Related Work

Transformers (Vaswani et al., 2017) are the foundation of many recent large language models (LLMs) that have achieved significant performance in a variety of 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 can be 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 prompt (Lester et al., 2021) and prefix (Li and Liang, 2021) tuning techniques, *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 being memory inefficient. 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, which should take the value of $1/r$, where r is the rank of the adapters (Zhang et al., 2023). $\boldsymbol{L}_1$ 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

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

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

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 the quantization of 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 training candidate architectures, either partially or completely (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). A benefit of the weight-sharing technique is that a super-network composed of a large number of subnetworks does not require additional memory and storage than 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

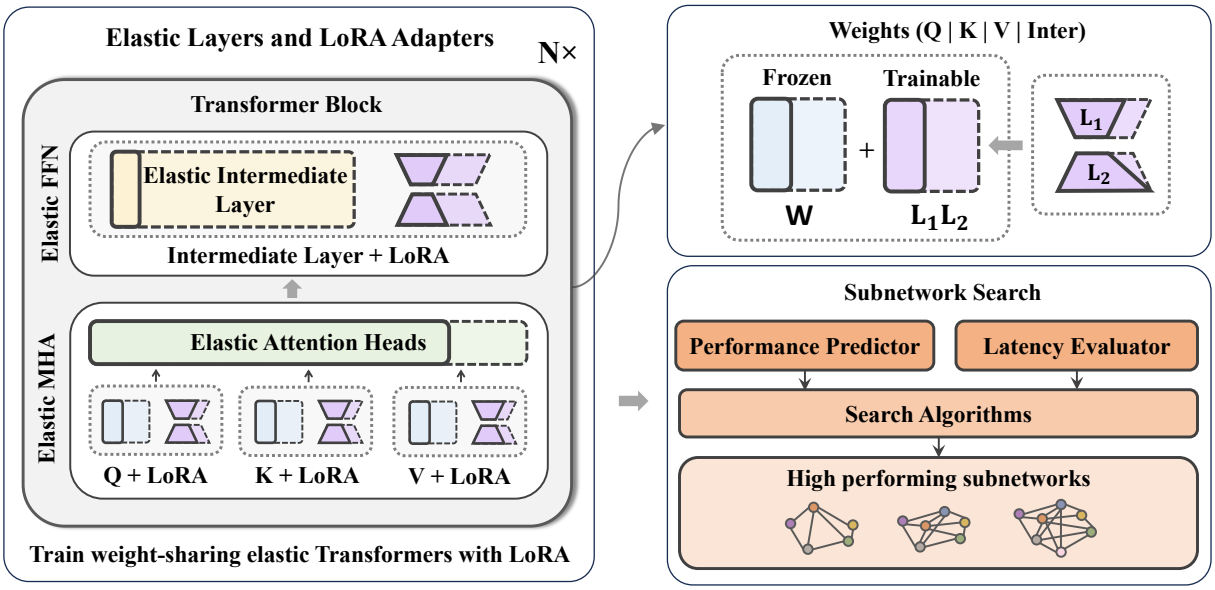

Figure 1: LoNAS end-to-end workflow. Low-rank adapters are attached to Transformer blocks. Elasticity is applied to the frozen layers in the original model and to the low-rank adapter.

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 at their dependent frozen weights. In the following sections, we describe LoNAS and present the results of generating super-networks for three language models.

3. Methodology

In this section, we 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 less storage and memory requirements. LoNAS end-to-end workflow is illustrated in Figure 1 and 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 tensors, resulting in linear projections using a subset of the original parameters of the layer's weights, $\boldsymbol{W}$. For our purposes, *elasticity* means that a selected layer, $\boldsymbol{l}$, can have multiple values for a particular property. For instance, if $\boldsymbol{l} \in \mathbb{R}^{m \times n}$, we might allow possible slices to create subnetworks in which $\boldsymbol{l}$ activates k of its columns, s.t., $k < n$, resulting in an active 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 directly account for the possible configurations of $\boldsymbol{W}$ 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 config-

urations. Memory consumption is reduced due to W remaining frozen. The only trainable parameters explored by LoNAS are the subsets of L_1 and 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 $W_\delta \subseteq W$ (Equation 4) is strictly dependent on the subset $L_{\delta 2} \subseteq L_2$. When choosing an elastic configuration, LoNAS maintains these structures with consistent shapes. L_1, on the other hand, can be sliced arbitrarily without the need to align with the frozen weights of the original model.

In the case of the multi-attention heads, LoNAS enables elasticity and the possibility to attach adapters to the **Q**, **K**, and **V** layers. We explore several configurations in Section 4. The original LoRA implementation only uses adapters in the **Q** and **V** layers of the attention heads, but experimentally, we have observed that in the case of weight-sharing super-networks, adding adapters to **K** improves the quality of the subnetworks.

As illustrated in Figure 1, LoNAS also enables the use of elastic adapters in layers of the Multilayer Perceptron (MLP) that follows each multi-head attention block. LoNAS enables elasticity either directly, on intermediate layers of the MLP or in the adapters attached to these layers. The goal in both cases is to obtain subnetworks with subsets of weights of the larger model. These efficient subnetworks have a reduced footprint compared to the original model. For example, in the case of GPT-J (Wang and Komatsuzaki, 2021), elasticity is applied to the **Fc-in** linear layer, while in the case of LLaMA (Touvron et al., 2023), elasticity is applied to the **Up** and/or the **Gate** linear projections.

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

Once elasticity has been enabled, resulting in a fixed search space of possible configurations for L_1 and L_2 at each of the selected layers, we need to pay particular attention to super-network training. At each iteration, we sample random subnetworks for each batch of data as recommended by Yu et al. (2020). During the forward pass, we need to take into account the frozen weights, W. 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

We conduct experiments by generating super-networks for multiple large language models and testing some of their subnetworks on six math reasoning datasets. The details of our setup and the analysis of the results are discussed next.

4.1. Experimental Setup

Datasets We compare our LoNAS results with those reported in the LLM-Adapters paper (Hu et al., 2023) for six math reasoning datasets: MultiArith (Roy and Roth, 2015), GSM8K (Cobbe et al., 2021), AddSub (Hosseini et al., 2014), AQUA (Ling et al., 2017), SingleEq (Koncel-Kedziorski et al., 2015), and SVAMP (Patel et al., 2021). In our experiments, we used the training data provided by the LLM-Adapters group, which combines all training datasets into a unified dataset that is used to train one general model and then test it on each dataset. Moreover, the LLM Adapters datasets are extracted with the help of Zero-shot *Chain of Thought (CoT)* (Wei et al., 2022) and GPT-3 text-davinci-003[1]. We also include these results in Table 1. Since publishing their results, the LLM-Adapters group has added new results in their Git repository[2] using a larger version of their training data. LoNAS produces competitive subnetworks that are also successful compared to their latest results.

Large Language Super-Networks We generate LoNAS super-networks for three foundational models: **LLaMA$_{7B}$** (Touvron et al., 2023), **GPT-J$_{6B}$** (Wang and Komatsuzaki, 2021) and **BLOOM$_{7.1B}$** (Scao et al., 2022). These autoregressive text generation models have a total of 6.7, 6.0, and 7.1 billion parameters, respectively. LLaMA$_{7B}$ was trained by Meta using data in 20 languages, but most of the text is in English. The data used for training LLaMA mostly comes from the CCNet (Wenzek et al., 2020), and C4 (Raffel et al., 2020b) datasets making up ≈82% of the training data. The rest of the data includes other sources like Github and Wikipedia. LLaMA$_{7B}$ has 32 layers and 32 heads in each multi-head attention layer. More details are available on the Hugging Face model card[3]. GPT-J$_{6B}$ was trained by Wang and Komatsuzaki (2021) using the English-only text dataset, the Pile (Gao et al., 2020). GPT-J$_{6B}$ has 28 Transformer blocks, 16 heads in each multi-head attention layer. More details are available on the Hugging Face model card[4]. BLOOM$_{7.1B}$ was trained by Scao et al. (2022) using the ROOTS corpus (Laurençon et al., 2022) with data on 59 languages. BLOOM$_{7.1B}$ has 30 Transformer blocks and 32 heads in each multi-head attention layer. More details are available in the Hugging Face model card[5].

[1] https://platform.openai.com/docs/models/gpt-3-5
[2] https://github.com/AGI-Edgerunners/LLM-Adapters
[3] https://huggingface.co/yahma/llama-7b-hf
[4] https://huggingface.co/EleutherAI/gpt-j-6b
[5] https://huggingface.co/bigscience/bloomz-7b1

Table 1: Efficiency (TFLOPs) and test accuracy (%) comparison of LoNAS with other LLM-Adapter approaches. LoNAS discovers alternative versions of LLMs with similar (**LoNAS-M**) or smaller number of total parameters (**LoNAS-H**) while maintaining or improving their accuracy. The results from GPT-3.5, S-Adapter, P-Adapter, and LoRA baseline are those reported by Hu et al. (2023). **LoNAS-M** is the maximal subnetwork in each super-network, and **LoNAS-H** is a subnetwork obtained with the heuristic described in Equation 7.

LLM	Method	TFLOPs	Total / Trainable Params	Datasets | % Accuracy						Average
				MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	
GPT-3.5	Zero-shot CoT	-	175B / 175B	83.8	56.4	85.3	38.9	88.1	69.9	70.4
$LLaMA_{7B}$	S-Adapter^{h}	1.7	6.7B / 200M	88.3	18.5	69.6	27.4	85.2	52.5	56.9
	S-Adapter^{p}	1.7	6.7B / 200M	88.3	18.5	69.6	15.6	79.4	52.0	53.9
	P-Adapter	1.7	6.7B / 200M	83.3	22.7	77.2	9.8	81.3	57.0	55.2
	LoRA	1.7	6.7B / **4.2M**	88.3	21.9	78.5	27.5	83.3	54.5	59.0
	LoNAS-M(Ours)	1.7	6.7B / 11.9M	**90.2**	**46.4**	88.1	33.5	**90.2**	**75.1**	70.5
	LoNAS-H(Ours)	**1.4**	**5.6B** / 11.9M	89.7	46.1	**88.6**	**36.6**	90.0	73.6	**70.8**
$GPT\text{-}J_{6B}$	S-Adapter^{h}	1.5	6.0B / 117M	82.5	4.5	55.7	3.9	67.6	39.5	42.3
	S-Adapter^{p}	1.5	6.0B / 176M	79.2	9.8	54.4	19.6	63.7	37.5	44.0
	P-Adapter	1.5	6.0B / 176M	79.2	11.0	65.8	11.8	69.6	44.5	47.0
	LoRA	1.5	6.0B / **3.7M**	79.2	10.6	69.6	2.0	71.6	45.0	46.3
	LoNAS-M(Ours)	1.5	6.0B / 8.2M	88.0	39.7	86.1	**38.9**	86.8	69.7	68.2
	LoNAS-H(Ours)	**1.2**	**5.1B** / 8.2M	**90.2**	**41.4**	**87.3**	38.6	**88.2**	**70.3**	**69.3**
$BLOOM_{7.1B}$	S-Adapter^{h}	1.8	7.1B / 125M	60.8	6.4	43.0	23.5	52.0	37.5	37.2
	S-Adapter^{p}	1.8	7.1B / 188M	70.6	8.3	50.6	13.7	50.0	35.5	38.1
	P-Adapter	1.8	7.1B / 125M	55.0	5.7	35.4	27.5	49.0	28.0	33.4
	LoRA	1.8	7.1B / **4.0M**	46.7	4.2	32.9	11.7	41.2	22.5	26.5
	LoNAS-M(Ours)	1.8	7.1B / 8.8M	**81.7**	30.6	76.2	31.5	76.4	60.8	59.5
	LoNAS-H(Ours)	**1.5**	**6.2B** / 8.8M	81.2	**33.2**	**78.7**	**34.6**	**78.1**	**62.3**	**61.3**

Subnetwork Search Once a super-network has been trained, we could use 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 select a subnetwork that improves on accuracy and efficiency. However, using NSGA-II is expensive when dealing with larger models. To quickly have an estimate of the quality of the smaller subnetworks, we use a heuristic in which from all the n possible configurations of an elastic layer, l_i, we select a configuration indexed at c, close to the middle of the range of possible configurations, formulated as

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

where **LoNAS-M** is the configuration of the maximal subnetwork, which is equivalent in its architecture to the original base model, and **LoNAS-H** is the subnetwork configuration obtained by the heuristic. Section 4.3 compares this heuristic to the evolutionary search and discusses the trade-offs in selecting each approach. To better illustrate the subnetwork search stage, Figure 2 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.

Alternative Adapters We compare LoNAS subnetworks with the models using several types of adapters reported by LLM-Adapters (Hu et al., 2023): **(1) S-Adapters^{h}** are placed after each of the main components of the Transformer block, i.e., after the multi-head attention (MHA) layer and the multi-layer perceptron (MLP). **(2) S-Adapters^{p}** are placed only after each MLP module. **(3) P-Adapters** are parallel adapters placed at the same level of the MHA or MLP layers. **(4) LoRA** are low-rank parallel adapters placed at the same level of the linear layers of the Transformer block while keeping the original weights, W, of the Transformer block frozen.

Implementation Details LoNAS is implemented as an extension of <*anonymized for review*>. We modified this library to enable hooks in the frozen weights and their alignment with the corresponding inserted low-rank adapters. We also patch Hugging Face's PEFT library[6] to enable elasticity at the low-rank adapters. We set the batch size to 16, the learning rate to 7×10^{-4}, and 16 epochs for training each super-network. The adapter target modules and the search spaces used for our main experiments are detailed in Table 5 of the Appendix. We also explore other search spaces that are described in Appendix B.

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

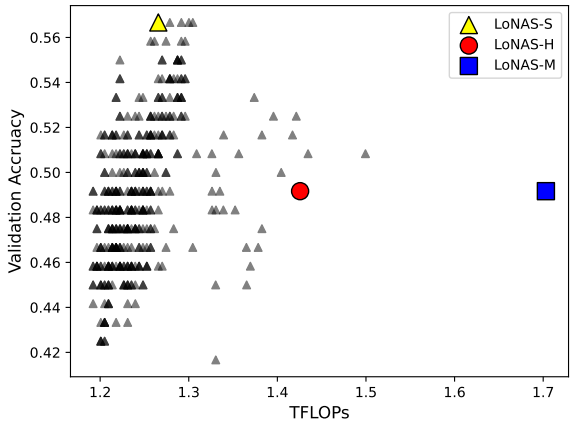

Figure 2: Search progression on the super-network trained with LLaMA$_{7B}$-LoNAS. Additional high-performing subnetworks are available from the Pareto frontier based on the desired performance in the accuracy/TFLOPs multi-objective space.

4.2. Overall Performance

Table 1 presents the results of a few selected sub-networks from each large language model super-network. All results for the baselines are obtained from Hu et al. (2023). **LoNAS-M** is the maximal subnetwork with the same architecture as the input pre-trained model with LoRA adapters while **LoNAS-H** corresponds to a smaller subnetwork obtained with the heuristic described in Equation 7. As shown in the table, the subnetworks obtained by LoNAS are competitive with the baselines. Whether **LoNAS-M** or **LoNAS-H**, there is a significant improvement in accuracy. The discovered subnetworks have fewer total parameters, although in some cases, they might have more trainable parameters than other adapter approaches. Most importantly, LoNAS' super-networks for LLaMA$_{7B}$, GPT-J$_{6B}$, and BLOOM$_{7.1B}$ all contain other high-performing subnetworks that are more efficient than the original pre-trained model, which means that LoNAS can obtain smaller and more efficient models with higher accuracy and speedups during inference. It is noteworthy that our LLaMA$_{7B}$-LoNAS super-network is competitive with GPT-3.5, which has 175B parameters, 26.12x more parameters than the LLaMA$_{7B}$-LoNAS-M subnetwork, and 31.82x more parameters than LLaMA$_{7B}$-LoNAS-H subnetwork. Additionally, both **LoNAS-M** and **LoNAS-H** outperform the updated SOTA results on the LLM-Adapters repository, which uses more training data.

4.3. Analysis of Subnetwork Evolutionary Search

As described in section 4.1, we devise a heuristic to quickly explore the quality of the subnetwork in our trained super-networks. From the above analysis, LoNAS can use the midpoint heuristic (Equation 7) to 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. To further explore our trained super-network, we apply the more advanced evolutionary search using the NSGA-II algorithm.

Figure 2 illustrates the search progression on the LLaMA$_{7B}$-LoNAS super-network. We use the accuracy on the validation set as a metric to guide the search process. After performing an evolutionary search, we select the smallest subnetwork with the highest accuracy, **LoNAS-E**, with 5.0B parameters and 1.26 TFLOPs compared to LLaMA$_{7B}$ with 6.7B parameters and 1.7 TFLOPs. As shown in Table 2, we can further obtain a smaller, more efficient fine-tuned model with higher accuracy compared with using LoRA, although the accuracy performance of **LoNAS-E** drops slightly compared to **LoNAS-M** and **LoNAS-H** in the test dataset. It is an interesting finding that the performance on the validation set does not match the performance on the test set, which implies the validation set does not provide accurate guidance for finding the subnetwork that performs the best on the test set. The reason we guess is that the validation set is randomly sampled from the unified dataset from the six individual math datasets, consequently, the distribution of datasets in the validation set has randomness. We believe that better results can be achieved if we use a validation set for each specific dataset. We will explore this idea in future work. We will explore this idea in future work.

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 120, and the number of subnetworks evaluated is 400. Each evaluation for one LLaMA$_{7B}$-LoNAS subnetwork took approximately 10 minutes, and completing the search stage took 2.7 days using a single GPU. 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 sub-network quality.

Table 2: Efficiency and accuracy comparison of three selected $LLaMA_{7B}$-LoNAS subnetworks. The validation set is randomly sampled from the unified training dataset. The test accuracy is the average accuracy on the six math reasoning datasets. Inference speedup is relative to the maximal subnetwork with the same configuration as the base model $LLaMA_{7B}$ and on 600 test samples from the MultiArith dataset. We use an Intel Xeon Platinum 8480+ with Advanced Matrix Extensions (AMX) enabled to collect the inference speedup in the CPU and a single NVIDIA Tesla V100 for the GPU. All subnetworks are in FP16 precision and KV cache is enabled.

Subnetwork	Search Method	TFLOPs	Total Params	Validation Accuracy(%)	Test Accuracy(%)	Inference Speedup	
						GPU	CPU
LoNAS-M	Maximal	1.70	6.7B	49.2	70.5	1.00×	1.00×
LoNAS-H	Heuristic	1.42	5.6B	49.2	**70.8**	1.16×	1.11×
LoNAS-E	Evolutionary	**1.26**	**5.0B**	**56.7**	68.4	**1.24**×	**1.14**×

Table 3: Accuracy (%) comparison of $LLaMA_{7B}$ + LoRA and $LLaMA_{7B}$ + LoNAS with the same trainable parameters.

Method	LoRA	LoNAS-M	LoNAS-H
TFLOPs	1.70	1.70	**1.42**
Params	6.7B / 4.2M	6.7B / 4.2M	**5.6B** / 4.2M
MultiArith	**88.3**	**88.3**	86.8
GSM8K	21.9	**31.4**	30.1
AddSub	78.5	**84.3**	84.1
AQuA	27.5	21.3	**29.9**
SingleEq	**83.3**	81.3	79.1
SVAMP	54.5	**68.5**	66.2
Average	59.0	62.5	**62.7**

4.4. LoRA vs. LoNAS with the Same Number of Trainable Parameters

As shown in Table 1, LoNAS applies more trainable parameters compared to LoRA (e.g., 11.9M vs. 4.2M) but produces subnetworks with a smaller total number of parameters, reducing the cost when using the models for inference. In both cases, the trainable parameters remain less than 1% of the total number of parameters in the base model. For a fair comparison, we set $LLaMA_{7B}$ + LoRA and $LLaMA_{7B}$ + LoNAS to have the same number of trainable parameters. In the case of LoNAS' super-network, low-rank elastic adapters were added to the **Q** and **V** modules of the Transformer block, similar to LLM-Adapters (Hu et al., 2023) to maintain the trainable parameters at 4.2M. As shown in Table 3, LoNAS outperforms LoRA under the same training parameter settings as well, confirming the effectiveness of LoNAS. Additionally, the comparisons in Table 1 and Table 3 reveal that injecting LoRA adapter into intermediate layers helps LoNAS train a more robust super-network with high-performing compressed subnetworks.

4.5. Inference Benefit Analysis of LoNAS

As shown in Table 1 and Table 2, LoRA provides benefits during training/fine-tuning since the weights 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 2, the subnetwork obtained using our heuristic, **LoNAS-H**, and the subnetwork obtained using evolutionary search, **LoNAS-E**, have fewer total parameters than the base model, which has the same architecture as subnetwork, **LoNAS-M**. The compressed subnetworks result in a speedup during inference of up to 1.24 times the inference time of the base model. In this experiment, we use a batch size of 1, a context window size of 2048 tokens, enable the KV cache, and set the precision of the weights to use FP16. To accelerate inference in CPU, we enable Intel's Advanced Matrix Extensions. Further speedup can be obtained by quantizing the discovered subnetworks (Section 5).

4.6. Ablation Study: Placement of Elastic LoRA Adapters in the Super-Network

LoRA adapters are often added only to the **Q** and **V** layers. In this section, we investigate how placing LoNAS' elastic adapters in non-traditional layers of the $LLaMA_{7B}$ super-network might affect the performance of its subnetworks. In addition to the **Q** and **V** layers, we explore adding LoNAS' elastic adapters to the **K** linear layers of the attention layers and the **Up** and **Gate** linear projections of the MLP in $LLaMA_{7B}$'s Transformer blocks. Table 4 shows the results with alternative placements for the additional elastic LoRA adapters. In general, despite the minor increase in cost related to the additional trainable parameters during training, adding more adapters in non-traditional layers improves the overall results. For instance, the subnetwork with the best average in all datasets is the one from a super-network with all possible adapter placements enabled, i.e., **LoNAS-H** in the

Table 4: Ablation studies for adapter target modules with $LLaMA_{7B}$ + LoNAS. We tried different placements for the elastic LoRA adapters and compared the maximal and heuristic subnetworks' accuracy.

LoRA Target Modules					Trainable%	Trainable Params	Subnetwork	Datasets | % Accuracy						Average
Q	K	V	Up	Gate				MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	
✓	✗	✓	✗	✗	0.062	4.2M	**LoNAS-M**	88.3	31.4	84.3	21.3	81.3	68.5	62.5
							LoNAS-H	86.8	30.2	84.1	29.9	79.1	66.2	62.7
✓	✓	✓	✗	✗	0.093	6.3M	**LoNAS-M**	86.2	33.7	85.1	28.0	83.7	70.9	64.6
							LoNAS-H	87.0	36.5	82.8	29.1	80.7	67.5	63.9
✓	✗	✓	✓	✗	0.119	8.1M	**LoNAS-M**	88.5	41.3	87.3	34.7	86.8	72.5	68.5
							LoNAS-H	88.2	43.3	86.8	35.8	85.6	72.7	68.7
✓	✓	✓	✓	✗	0.151	10.2M	**LoNAS-M**	**90.8**	42.8	87.3	34.7	87.0	73.6	69.4
							LoNAS-H	89.2	44.1	88.1	**41.7**	88.0	72.2	70.6
✓	✗	✓	✓	✓	0.177	11.9M	**LoNAS-M**	90.2	**46.4**	88.1	33.5	**90.2**	**75.1**	70.5
							LoNAS-H	89.7	46.1	**88.6**	36.6	90.0	73.6	70.8
✓	✓	✓	✓	✓	0.208	14.0M	**LoNAS-M**	89.7	45.8	88.4	37.0	89.4	74.8	70.8
							LoNAS-H	89.8	45.9	88.1	37.8	89.6	74.1	**70.9**

last group of the table. Table 4 is sorted in ascending order of the number of trainable parameters, giving us insights into the additional costs required to make the super-network more robust.

5. Discussion and Future Work

Quantization of the Subnetworks LoNAS efficient subnetworks in FP16 precision improve inference time and allow their deployment to smaller systems. Other compression techniques can be applied to the discovered subnetworks to obtain additional improvements in their inference efficiency. For instance, selected subnetworks can be quantized to continue reducing their memory footprint and improving their inference efficiency at deployment.

Quantization of the Frozen Weights Quantization can also benefit the training efficiency of LoNAS. 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 to further reduce memory consumption (Equation 8). QLoRA also quantizes the quantization constants, c_i which further reduces the memory footprint during training. QLoRA can be formulated as

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

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

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, but it is in our future plans to further improve the efficiency of LoNAS. Quantization and other techniques will help LoNAS continue reducing its footprint during training, as done by QLoRA.

Weight reordering Weight reordering strategies are often used in weight-sharing NAS to improve the quality of the super-networks. This step can be very expensive 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. The code will be released as open-source, and we provide examples for the reviewers.

6.1. Ethics Statement and Limitations

Large language models (LLMs) are relatively new technologies plagued 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 have a negative impact outside of research. These limitations are outside the scope of this paper. LoNAS 's goal is to deliver compressed models that can run efficiently on resource-constrained devices without or with minor degradation in accuracy. However, we believe that the limitations mentioned above (and others omitted here) must be taken into account when designing systems and applications that use large-language models, and this is a concern that we will incorporate into our future research. Before deploying these models, it is imperative to conduct exhaustive testing and identify risks and vulnerabilities to prevent any potential harm.

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.

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*.

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.

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

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

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.

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 Jeff Dean. 2015. Distilling the knowledge in a neural network.

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.

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

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.

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.

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

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*.

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.

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

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. 2020. CCNet: Extracting high quality monolingual datasets from web crawl data. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 4003–4012, Marseille, France. European Language Resources Association.

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

Jiahui Yu 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.

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. Search Spaces used in the Main Experiments in the Paper

Table 5 describes the search spaces used for our main experiments in the paper, with their results described in Table 1. Future versions of LoNAS will investigate approaches for better designing the search space to obtain more robust supernetworks with better performant subnetworks.

B. Increasing the Complexity of the Search Spaces

An open problem in research in Neural Architecture Search has to do with the effective design of search spaces. This section explores various possible values for the width of the elastic layers used in LoNAS experiments. Table 6 includes the possible configurations explored that lead to additional search spaces, and Table 7 presents the results of the maximal subnetwork, **LoNAS-M**, and the subnetwork obtained with the heuristic, **LoNAS-H**, for each of the possible search spaces. Search Space ID 1 corresponds to the search space used in the main experiments of the paper. In future work, we plan to conduct the expensive search stage to discover other high-performing subnetworks in each search space.

Table 5: Search spaces and target modules used in the main experiments For each module, the number in the search space represents the possible output hidden size of the linear module. LoRA_A and LoRA_B correspond to $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in Equations 5 and 6 respectively.

LLM	Adapter Target Modules	Search Space
LLaMA_{7B}	Q, V, Up, Gate	Q & K & V & Q-LoRA_B & V-LoRA_B: [4096, 3072] Q-LoRA_A & V-LoRA_A: [8, 6] Up & Gate & Up-LoRA_B & Gate-LoRA_B: [11008, 8256, 5504] Up-LoRA_A & Gate-LoRA_A: [8, 6]
GPT-J_{6B}	Q, V, Fc-in	Q & K & V & Q-LoRA_B & V-LoRA_B: [4096, 3072] Q-LoRA_A & V-LoRA_A: [8, 6] Fc-in & Fc-in-LoRA_B: [16384, 12288, 8192] Fc-in-LoRA_A: [8, 6]
$\text{BLOOM}_{\text{7.1B}}$	Q, K, V, Dense	Q & K & V & Q-LoRA_B & K-LoRA_B & V-LoRA_B: [4096, 3072] Q-LoRA_A & K-LoRA_A & V-LoRA_A: [8, 6] Dense & Dense-LoRA_B: [16384, 12288, 8192] Dense-LoRA_A: [8, 6]

Table 6: Additional search spaces for the LoNAS + LLaMA_{7B} super-network.

Search Space ID	Q & K & V & Q-LoRA_B & V-LoRA_B	Q-LoRA_A & V-LoRA_A	Up & Gate & Up-LoRA_B & Gate-LoRA_B	Up-LoRA_A & Gate-LoRA_A
1	[4096, 3072]	[8, 6]	[11008, 8256, 5504]	[8, 6]
2	[4096, 3072, 2048]	[8, 6]	[11008, 8256, 5504]	[8, 6]
3	[4096, 3584, 3072]	[8, 6]	[11008, 8256, 5504]	[8, 6]
4	[4096, 3584, 3072, 2560]	[8, 6]	[11008, 8256, 5504]	[8, 6]
5	[4096, 3072]	[8, 6]	[11008, 9632, 8256, 6880, 5504]	[8, 6]
6	[4096, 3072]	[8, 6, 4]	[11008, 8256, 5504]	[8, 6, 4]
7	[4096, 3072]	[16, 12]	[11008, 8256, 5504]	[16, 12]
8	[4096, 3584, 3072]	[8, 6]	[11008, 9632, 8256, 6880, 5504]	[8, 6]

Table 7: Efficiency (TFLOPs) and test accuracy (%) comparison of LoNAS + LLaMA_{7B} with different search spaces (Table 6). **LoNAS-M** is the maximal subnetwork in each super-network, and **LoNAS-H** is a subnetwork obtained with the heuristic described in Equation 7.

Search Space ID	Method	TFLOPs	Total/Trainable Params	Datasets & % Accuracy						Average
				MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	
1	**LoNAS-M**	1.70	6.7B / 11.9M	90.2	46.4	88.1	33.5	90.2	75.1	70.5
	LoNAS-H	1.42	5.6B / 11.9M	89.7	46.1	88.6	36.6	90.0	73.6	70.8
2	**LoNAS-M**	1.70	6.7B / 11.9M	88.2	40.6	86.1	**40.6**	87.0	70.9	68.9
	LoNAS-H	**1.29**	**5.1B** / 11.9M	87.0	41.5	87.1	34.3	85.8	69.2	67.5
3	**LoNAS-M**	1.70	6.7B / 11.9M	89.7	45.0	89.1	33.9	90.4	74.9	70.5
	LoNAS-H	1.36	5.3B / 11.9M	89.2	45.8	88.6	37.8	88.6	73.8	70.6
4	**LoNAS-M**	1.70	6.7B / 11.9M	89.7	43.6	87.1	33.9	89.4	73.7	69.6
	LoNAS-H	1.36	5.4B / 11.9M	89.7	44.7	87.6	37.0	87.8	72.8	69.9
5	**LoNAS-M**	1.70	6.7B / 11.9M	90.5	47.1	88.9	37.4	**90.9**	**75.8**	**71.8**
	LoNAS-H	1.42	5.6B / 11.9M	88.8	45.9	87.8	39.4	88.6	74.1	70.8
6	**LoNAS-M**	1.70	6.7B / 11.9M	89.3	44.7	88.4	35.4	89.8	73.7	70.2
	LoNAS-H	1.42	5.6B / 11.9M	89.8	43.8	87.3	34.6	86.8	73.5	69.3
7	**LoNAS-M**	1.70	6.7B / 11.9M	**91.3**	**47.4**	**89.4**	37.4	90.2	74.9	**71.8**
	LoNAS-H	1.42	5.6B / 24.0M	**91.3**	46.7	88.1	39.8	89.8	74.7	71.7
8	**LoNAS-M**	1.70	6.7B / 11.9M	90.8	**47.9**	**90.1**	37.8	89.6	74.8	**71.8**
	LoNAS-H	1.36	5.3B / 11.9M	90.3	46.6	88.4	39.0	88.4	73.4	71.0