STACKING SMALL LANGUAGE MODELS FOR GENERALIZABILITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances show that large language models (LLMs) generalize strong performance across different natural language benchmarks. However, the large size of LLMs makes training and inference expensive and impractical to run in resource-limited settings. This paper introduces a new approach called fine-tuning stacks of language models (FSLM), which involves stacking small language models (SLM) as an alternative to LLMs. By fine-tuning each SLM to perform a specific task, this approach breaks down high level reasoning into multiple lower-level steps that specific SLMs are responsible for. As a result, FSLM allows for lower training and inference costs, and also improves model interpretability as each SLM communicates with the subsequent one through natural language. By evaluating FSLM on common natural language benchmarks, this paper highlights promising early results toward generalizable performance using FSLM as a cost-effective alternative to LLMs.

1 INTRODUCTION

Since the publication of the transformer paper Vaswani et al. (2017), a considerable amount of research devoted to large language models (LLMs) has shown that LLMs are capable of generalizing well on natural language benchmarks and that new emergent properties appear as LLMs increase in scale. Devlin et al. (2019); Wei et al. (2022). LLMs seem to follow some empirical scaling laws, where larger datasets, compute and model size contribute to improvements in model performance. Kaplan et al. (2020)

As language models and datasets increase in size, a growing need emerges to identify methods to run language models in resource-limited settings where large amounts of compute are inaccessible. In fact, multiple methods have been documented and researched in recent years to make LLM training or inference more computationally efficient. One such method is fine-tuning: given a pre-trained model, fine-tuning that model for specific tasks can cause that model to score better on benchmarked tasks downstream. Brown et al. (2020) Furthermore, more efficient methods of fine-tuning such as LoRA and QLoRA also show that adding a trainable adapter to LLMs whose weights are frozen also allows for faster fine-tuning while showing strong signs of solid model performance. Hu et al. (2021); Dettmers et al. (2023)

Additionally, recent work indicates that small language models (SLM), such as Microsoft's Phi-3, can still achieve decent performance on natural language benchmarks. This finding is important, as it suggest that small language models, which are a few orders of magnitude smaller than state-of-the-art LLMs, can still achieve solid performance on various benchmarks. Abdin et al. (2024)

This paper aims to build on both the fine-tuning and small language model directions, in order to identify methods that allow for cost-effective training and inference in resource-limited settings. As a result, this paper proposes a new model framework called Fine-tuning Stacks of Language Models (FSLM) - or "stacking" - which involves chaining multiple specialized small language models together such that the framework's input and output resemble those of performant language models.

FSLM takes loose inspiration from the human brain, where different components specialize in different tasks. For small language models, because each SLM has limited capabilities due to its small size, FLSM aims to fine-tune each SLM to specialize in a specific task. As a result, the motivating question becomes: how small can the SLMs be, such that the fine-tuned stack of SLMs is still capable of generalizing on various natural language benchmarks?

Our work challenges the lower-bound for SLM size by evaluating an FSLM stack of four Pythia models of 160 million parameters each. Biderman et al. (2023) By fine-tuning this FSLM stack on the Alpaca dataset, and benchmarking FSLM and models of similar size, this paper shows that FSLM stacks show promise as lightweight alternatives to heavier LLMs.

Thus, this paper's contributions can be summarized as:

- Proposing the FSLM stack as a lightweight framework to evaluate small language models in resource-limited settings.
- Introducing model distillation to fine-tune SLMs in order to minimize the need for human supervision or labeling.
- Identifying early signs of FSLM generalizability by comparing FSLM of Pythia-160M models with Pythia and Flan models of comparable sizes
- Documenting model explainability by looking at the intermediary outputs between SLMs in the FSLM stack.

2 Related Work

2.1 Model Fine-Tuning

In recent years, researchers have shown that pre-training a language model in a self-supervised fashion, followed by fine-tuning that same model to a variety of tasks, improves model performance downstream on natural language benchmarks. OpenAI's GPT is a notable example of fine-tuning a pre-trained model. Brown et al. (2020) Because fine-tuning entire models is expensive, researchers have developed different methods to minimize computational cost while still achieving similar model performance.

Hu et al. (2021) introduced **Low-Rank Adaptation (LoRA)** as a fine-tuning approach. LoRA freezes the weights of the original pre-trained model, and adds an "adapter" component, located between the original model output and the actual text output. Instead of the adapter being a fully connected layer, the adapter uses matrix factorization to generate low-rank matrix multiplications that approximate the fully connected equivalent. Low-rank matrix multiplication, however, is less computationally expensive than running inference on a fully connected layer. Hu et al. (2021) then show that LoRA can maintain or even improve model performance. Dettmers et al. (2023) developed **QLoRA**, which performs quantization to further improve LoRA. Both QLoRA and LoRA are considered to be **Parameter-Efficient Fine-Tuning (PEFT)** methods, a group of methods that aim to increase the efficiency of fine-tuning models. Xu et al. (2023)

2.2 Model Compression

Model compression techniques aim to either shrink a given model's size, or to train a smaller model to learn from a larger one.

For instance, **quantization** reduces the precision of the model weights, thus decreasing the overall size of the model. Even though the model loses precision, if quantization is implemented correctly, the model should maintain a similar level of performance while experiencing a speedup for training and inference. Jacob et al. (2017)

Model pruning removes weights whose values are close to zero, thus eliminating weights that may not be contributing to the model's main inference. Cheng et al. (2024)

Model distillation is another method of interest: using a teacher-student architecture, a smaller "student" model learns from a larger "teacher" model that should be already well-trained. As a result, the teacher model distills its internal knowledge to the student model, by providing the student model inputs and outputs to learn from during this training process. Hinton et al. (2015); Sanh et al. (2020)

3 METHOD

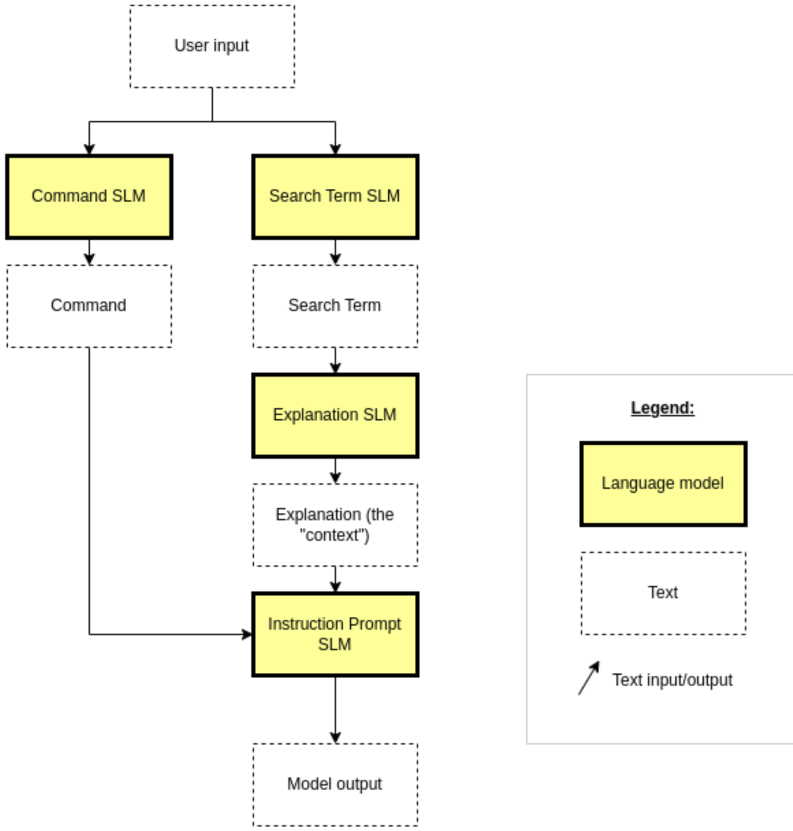

Figure 1: A visual representation of the FSLM stack.

3.1 FSLM METHOD OVERVIEW

The FSLM framework consists of four small language models (SLM) that each specialize in a specific task, as shown in Fig. 1. A human user would supply a prompt to the FSLM framework, and the FSLM framework responds with a textual output. Internally, the SLMs look for specific textual elements from either the user's input or another SLM's output. As a result, each individual SLM is compensating for its limited capabilities by instead specializing in a specific task. As a result, the overall framework follows an information flow where textual information is slowly processed towards the intended model output.

3.2 CHOICE OF MODELS

We use the Pythia 160M GPT-NeoX architecture from the Pythia suite, as Pythia allows for ease of future scalability as we can evaluate on different model sizes. Biderman et al. (2023) Pythia also integrates well with LM-eval, which we use to evaluate FSLM on natural language benchmarks. Gao et al. (2024)

3.3 CHOICE OF DATASET

We use the Alpaca dataset to train FSLM in an instruction-tuning manner. Taori et al. (2023) Alpaca contains 52,000 self-instruct generated instructions covering a wide array of applications. As of this writing, we selected a subsample of 5,000 instructions to fine-tune FSLM.

3.4 Training Data Generation

In order to properly distill the intermediary texts between SLMs, we use the Llama 3.2 (3B) model to generate texts, a recent addition to the Llama family of LLMs. Touvron et al. (2023)

3.5 Fine-Tuning

We use HuggingFace's PEFT implementation to run LoRA for fine-tuning.

4 Experiments

4.1 Natural Language Benchmarks

We use Eleuther AI's LM-Evaluation Harness to run natural language tasks from TinyBenchmarks. Gao et al. (2024); Polo et al. (2024)

Model	tinyArc	tinyMMLU
FSLM (4x Pythia-160M)	0.3349	0.3208
Pythia-160M (no adapter)	0.3213	0.3014
Pythia-1B (no adapter)	0.2945	0.2720
Flan-T5-Base (250M) (no adapter)	0.2781	0.3615
Flan-T5-Large (780M) (no adapter)	0.4209	0.4415

Table 1: Natural language benchmark results. All tasks are zero-shot, accuracy is the scoring metric. All Pythia models are taken from step 130,000.

From Table 1, we observe that our FSLM stack (following fine-tuning) performs better than non-adapter 160M and 1B Pythia models on tinyArc and tinyMMLU. This shows that fine-tuning specialized models in a ”stack” does not worsen overall model performance compared to vanilla Pythia models of comparable size - rather, FSLM actually observes an increase in performance relative to Pythia models.

Even though our FSLM implementation performs better than Google's Flan-T5-Base on tinyArc, Flan-T5-Base's performance on tinyMMLU is higher than FSLM's. Notably, Flan-T5-Large outperforms FSLM on both tasks by a noticeable margin. While FSLM on Pythia-160M shows encouraging early signs, it's possible that Flan-T5 models may exhibit superior performance due to their pre-training or fine-tuning processes. Chung et al. (2022) As a result, it becomes relevant to use different pre-trained models for FSLM and to run fine-tuning on different datasets as future experiments to implement.

4.2 Qualitative Analysis for Model Interpretability

Our FSLM implementation with four Pythia-160M is capable of simple question and answering in a coherent manner, as shown in Table 2. Since our FSLM framework (approximately 640M parameters, or around 1.4 GB) is sufficiently small to run on most mobile phones or personal computers, the coherence and natural-sounding response further show promise that FSLM can run in resource-limited settings and exhibit human-like responses.

The intermediary outputs of SLMs within FSLM is of particular interest, because these responses allow us to directly evaluate model interpretability. Accordingly, we observe in Table 2 that the intermediary SLM outputs match very strongly with each pre-defined task, at least from the perspective of a human observer. While this shows that LoRA fine-tuning for FSLM is cost-effective, these intermediate SLM responses also serve as a checkpoint to flag potential mistakes or hallucinations. Because each SLM is specialized for a specific task, we expect the scope of the responses for each SLM to be somewhat bounded. As a result, if we detect that one of the responses seems wrong, it may be sufficient to only re-tune that single SLM, instead of the whole FSLM stack. In addition to promoting model explainability, this design would also minimize compute costs needed to fix overall model performance throughout model deployment.

Section	Text
User input	**What are ways to stay healthy?**
Instruction	Find a healthy diet.
Search term	”How to keep your body healthy”
Explanation (the ”context”)	The most important thing to remember is to keep your body healthy. If you don’t, your body will be damaged and you’ll be unable to perform the essential tasks of daily living.
Final output	**Healthy eating is a key component of maintaining a healthy lifestyle.**

Table 2: Sample breakdown of the intermediary texts generated by FSLM.

However, throughout our model development process, we observed that FSLM responses can vary from one inference call to the next. As a result, future work should investigate optimal model temperature and top-k and top-p values in order to ensure repeatability and minimize high variances in model responses.

5 Conclusion and Discussion

The objective of this paper was to evaluate whether FSLM, a stack of task-specific SLMs, can perform well on natural language benchmarks and also exhibit natural-sounding text responses. By running natural language benchmarks, we determined that there were promising signs showing that FSLM’s Pythia models perform on par with vanilla Pythia models of comparable sizes, suggesting that stacking fine-tuned specialized models can lead to accurate models at small scales. Additionally, by observing the full response of a sample model output, we determined that the final output was coherent and natural-sounding, and that the intermediary outputs were also highly aligned to each SLM’s intended task. Additionally, FSLM’s modular design could allow for easy model debugging and replacement of faulty SLMs. These results demonstrate encouraging signs that stacks of highly specialized small language models can perform as well as equivalent models of the same size, making FSLM architectures a potential area of interest for resource-limited compute settings.

One main limitation concerns the limited scope for natural language benchmark evaluations. Because FSLM is a new implementation, we needed to write additional code to integrate it with existing lm-eval tasks, which initially limited the scope of tasks we could run as of this writing. Consequently, future work should increase the number of natural language benchmarks, and also evaluate model perplexity for token generation, and rouge scores for model summarization. Furthermore, surveys with human observers interacting with FSLM would be beneficial, as we would be able to quantitatively assess the quality and helpfulness of human-to-model interactions.

Another limiting factor is the fine-tuning scope. Future work should try different fine-tuning datasets and determine to what extent dataset quality influences model performance downstream. On a similar topic, model pre-training should also be documented, as shown by the flan-T5 models’ superior performances. Future work should investigate fine-tuning SLMs across different architectures that underwent different pre-training processes.

6 Reproducibility Statement

All the code used in this paper is accessible publicly on GitHub. The code is written in Jupyter Notebooks, which makes it easy for researchers to run and reproduce these results. Due to the double-blind submission, the Github link is not displayed here, though the codebase is available upon request.

REFERENCES

Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024. URL https://arxiv.org/abs/2404.14219.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling, 2023. URL https://arxiv.org/abs/2304.01373.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.

Hongrong Cheng, Miao Zhang, and Javen Qinfeng Shi. A survey on deep neural network pruning-taxonomy, comparison, analysis, and recommendations, 2024. URL https://arxiv.org/abs/2308.06767.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models, 2022. URL https://arxiv.org/abs/2210.11416.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023. URL https://arxiv.org/abs/2305.14314.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL https://zenodo.org/records/12608602.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference, 2017. URL https://arxiv.org/abs/1712.05877.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples, 2024. URL https://arxiv.org/abs/2402.14992.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/1910.01108.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.

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. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models, 2022. URL https://arxiv.org/abs/2206.07682.

Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL https://arxiv.org/abs/2312.12148.