Improving INSTRUCT Models for Free: A Study on Partial Adaptation

Anonymous ACL submission

Abstract

Instruct models, obtained from various instruction tuning or post-training steps, are commonly deemed superior and more usable than their base counterpart. While the model gains instruction following ability, instruction tuning may lead to forgetting the knowledge from pre-training or it may encourage the model being overly conversational or verbose. This, in turn, can lead to degradation of in-context few-shot learning performance. In this work, we study the performance trajectory between base and instruct models by scaling down the strength of instruction-tuning via the partial adaption method. We show that, across several model families and model sizes, reducing the strength of instruction-tuning results in material improvement on a few-shot in-context learning benchmark covering a variety of classic natural language tasks. This comes at the cost of losing some degree of instruction following ability as measured by AlpacaEval. Our study shines light on the potential trade-off between in-context learning and instruction following abilities that is worth considering in practice.

1 Introduction

Training Large Language Models (LLMs) involves multiple steps, broadly categorized into pre-training and post-training. In pre-training, the *base* model acquires the bulk of its knowledge through the next-token prediction objective. Post-training usually involves supervised fine-tuning (SFT) and multiple rounds of reinforcement learning from human feedback (RLHF), resulting in an *instruct* model that is better at following instructions and more aligned with user goals.

However, both SFT and RLHF, to some degree, encourage the model to produce long and conversational responses. This may be an unwanted feature when testing on extractive and/or structured natural language processing (NLP) tasks such as classification, name entity recognition, or extractive question answering. In these cases, the responses need to be concise and exact, and any additional chattiness creates issues in parsing the responses. Before instruct models became available, this need was fulfilled decently by the emergent few-shot in-context learning (ICL) abilities of the base model (Wei et al., 2022). Few previous studies touch on the pros and cons of base and instruct models. One example is Cuconasu et al. (2024) which shows how base models work better than instruct models on RAG-related tasks.

Our work aims to fill this gap and thoroughly explores the performance trajectory between base and instruct models. In order to study the learning dynamics between base and instruct models, we need access to the model checkpoints saved during instruct tuning, which are rarely available, especially for best performing open-weight models. Therefore as a surrogate of this (Na et al., 2024), we resort to a simple training-free technique, *partial adaptation (or PAd)* (Fleshman and Van Durme, 2024), to scale the instruction-tuning strength in a post-hoc manner. Concretely, we create in-between models by partially adapting the base model (with weights $\mathbf{W_B}$) to instruct (with weights $\mathbf{W}_I$): M_λ with weights $\mathbf{W_B} + \lambda\mathbf{A}$ $(\lambda \in [0, 1])$ where $\mathbf{A} \equiv \mathbf{W}_I - \mathbf{W}_B$. Hence, M_0 is the base model and M_1 is the instruct model (see Section 2 for more details).

Using 18 open-weight LLMs, we evaluate these partially adapted models on a benchmark containing 21 classic NLP tasks using few-shot in-context learning. We find that, for all models, the best performance is always achieved when $\lambda < 1$, i.e., when instruction tuning strength is scaled down. And the optimal choice of λ leads to a few percent points improvement with respect to both the base and instruct models.

However, perhaps not surprisingly, we also find that once evaluated on an instruction following benchmark, AlpacaEval 2.0 (Dubois et al., 2024),

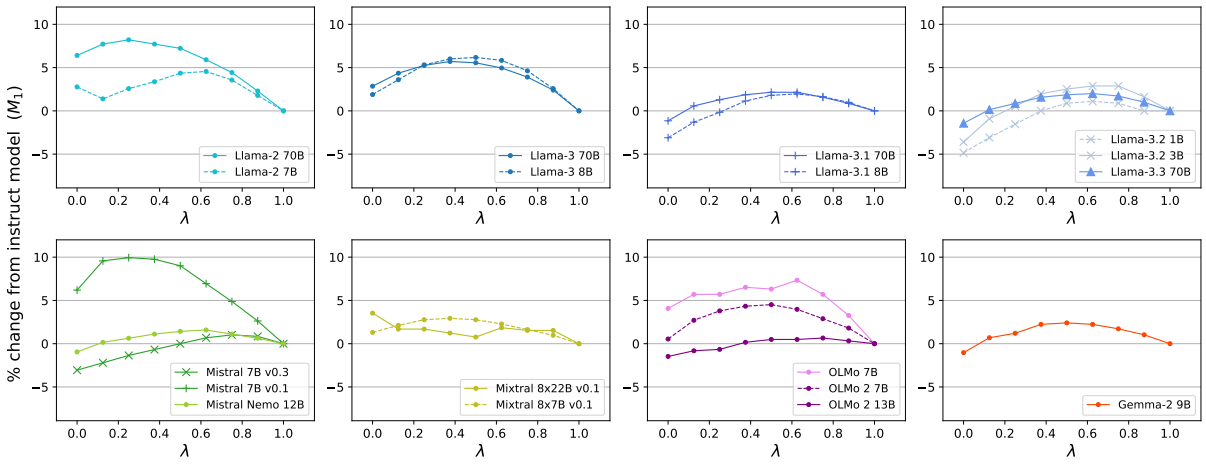

Figure 1: Performance on the in-context learning benchmark: fractional difference (percent value) between the performance of each partially adapted model M_λ and the instruct baseline M_1 for all the models we have tested.

the best partially adapted models selected by the ICL benchmark consistently under-perform their fully instruction tuned counterparts. Nonetheless, especially for models of larger sizes, we can oftentimes find a $\lambda < 1$, for which the AlpacaEval performance shows little to no drop, yet there is still a gain in the ICL benchmark.

In summary, through this comprehensive analysis, we demonstrate that the best ICL model is not necessarily the instruct model. We believe partial adaptation represents a training-free yet effective option worth exploring when dealing with ICL tasks that are structured, more extractive in nature, or requiring shorter answers. We hope our study highlights the opportunities and can inspire future work in better understanding the learning dynamics in LLM post-training.

2 Preliminary: Partial Adaptation

Fleshman and Van Durme (2024) propose that the contribution of LLM post-training can be isolated by simply differencing the weights of the instruct and base model, $\mathbf{A} \equiv \mathbf{W}_I - \mathbf{W}_B$. $\mathbf{A}$ can be seen as an adapter to be applied on top of the base model and the strength of the adapter can be adjusted in the form of $\mathbf{W_B} + \lambda\mathbf{A}$ ($\lambda \in [0, 1]$). This technique is called *partial adaptation* (PAd), with the implied meaning as *partially adapting the base model to instruction following*. In fact, in one single experiment, Fleshman and Van Durme (2024) also showed that partial adaptation leads to improvement on a zero-shot QA task to support their conjecture that instruction-tuning likely degrades knowledge from pretraining. We are inspired by this observation and conduct thorough analysis across models and datasets in this paper.

The partially adapted model can also be viewed as the weighted average between base and instruct models. Hence, we consider a new model M_λ with weights $(1-\lambda)\mathbf{W}_B + \lambda\mathbf{W}_I$, so that M_0 and M_1 correspond to the base and instruct models respectively. Open-weight models that we consider are listed in Table 1.[1] In practice, we enumerate λ from $\{0, \frac{1}{8}, \frac{2}{8}, \frac{3}{8}, \frac{4}{8}, \frac{5}{8}, \frac{6}{8}, \frac{7}{8}, 1\}$.

3 Evaluation Benchmarks

We evaluate partially adapted models on two benchmarks for testing ICL and instruction following performance respectively.

3.1 In-Context Learning Benchmark

Our primary goal is to measure performance on few-shot in-context learning. We assemble a benchmark of various classic NLP tasks to test a variety of natural language abilities. The composition of the benchmark is shown in Table 2 and described in details in Appendix A.1. We particularly include tasks from the financial domain because classic structured NLP tasks (classification, name entity recognition, extractive QA) widely appear in financial data analysis. Each dataset is tested in a few-shot manner, where the number of shots is displayed in Table 2. Shot selection is random and

[1] For all of the models, except Mixtral 8×22B, the embedding lookup tables of the base and instruct versions are aligned, so merging is straightforward. For Mixtral 8×22B, there are additional special tokens in the vocabulary of the instruct model. We take care of this by applying $\lambda = 1$ for those weights that are only present in the instruct model.

done independently for each example.

Depending on the dataset, evaluation proceeds in one of three possible ways (more details in Appendix A.2). For multiple choice (MC) datasets, we use the model to score each of the possible answers using likelihood and pick the highest ranking one. As a variation of this, fast multiple choice (FMC), instead of scoring each response, the model is prompted with them as a bulleted list (in MMLU format (Hendrycks et al., 2021)) and only the individual tokens corresponding to the bullets (A, B, C, ...) are scored and ranked. Finally for generation (G) datasets, the model generates a completion which is then parsed and compared to the ground truth answer.[2]

When a single dataset is evaluated in multiple ways (different prompts or different evaluation styles: MC vs. FMC vs. G), we aggregate these individual scores by taking their maximum. All metric sores are in a scale of 0 to 100. Therefore, we are able to *average* dataset-level scores into one single model-level score. More details about the templates and metrics that we use in our evaluation protocol are presented in Appendix A.3 and A.4.

3.2 AlpacaEval

Instruction following is a broad concept. In this work, we refer to it as the model's ability to answer open-ended questions from users, as exemplified by Chatbot Arena (Chiang et al., 2024).[3] Here, we test on AlpacaEval 2.0 (Dubois et al., 2024), which has a Spearman correlation of 0.98 with Chatbot Arena while being cost-efficient. For each value of λ, we obtain the length-controlled win-rate of M_λ against GPT-4 Preview (11/06) (Li et al., 2023) judged by GPT-4o.[4]

4 Results

Figure 1 and Figure 2 illustrate the relative performance change of each partially adapted model M_λ against the instruct model M_1 on ICL and and AlpacaEval benchmark, respectively. And Figure 4 and Figure 3 in Appendix B shows the corresponding absolute values. We summarize the absolute performance of base/instruct models and the best partially adapted models as well as the best λ^* in Table 1.

Model	Base/Inst.	Best$^{\lambda^*}$	δ_{wr}
Llama-2 7B	51.9/50.5	$52.8^{5/8}$	−4.35
Llama-2 70B	64.8/60.9	$65.9^{2/8}$	−16.64
Llama-3 8B	59.4/58.3	$61.9^{4/8}$	−15.81
Llama-3 70B	68.5/66.6	$70.4^{3/8}$	−6.02
Llama-3.1 8B	59.3/61.2	$62.4^{5/8}$	−5.58
Llama-3.1 70B	69.0/69.8	$71.3^{4/8}$	−5.30
Llama-3.2 1B	43.2/45.4	$45.9^{5/8}$	−8.93
Llama-3.2 3B	53.6/55.6	$57.2^{5/8}$	−8.89
Llama-3.3 70B	69.0/70.0	$71.4^{5/8}$	−0.93
Mistral 7B v0.1	56.6/53.3	$58.6^{2/8}$	−6.73
Mistral 7B v0.3	57.1/58.9	$59.5^{6/8}$	−1.57
Mistral Nemo 12B	62.5/63.1	$64.1^{5/8}$	−5.70
Mixtral 8x7B v0.1	62.2/61.4	$63.2^{3/8}$	−14.48
Mixtral 8x22B v0.1	67.4/65.1	$67.4^{0/8}$	NA
Gemma-2 9B	57.6/58.2	$59.6^{4/8}$	−6.52
OLMo 7B 0724	51.1/49.1	$52.7^{5/8}$	−6.79
OLMo 2 7B 1124	55.7/55.4	$57.9^{4/8}$	−7.41
OLMo 2 13B 1124	60.2/61.1	$61.5^{6/8}$	−3.98

Table 1: For each of the models (LLama-2 (Touvron et al., 2023), Llama-3 (Dubey et al., 2024; Meta, July 2024,S,D), Mistral (Jiang et al., 2023), Mistral-NeMo (mistral.ai, July 2024), Mixtral (Jiang et al., 2024; mistral.ai, April 2024), Gemma-2 (Riviere et al., 2024), OLMo (Groeneveld et al., 2024), and OLMo-2 (OLMo et al., 2024)) in the first column we report the base and instruct baseline performance on the benchmark, together with the best performance obtained by varying λ and the best value λ^* at which peak performance is achieved. The last columns reports the *absolute* change in win rate for the best PAd model with respect to the instruct version as determined by AlpacaEval 2.0. NA is because $\lambda^* = 0$ and we don't evaluate AlpacaEval on the base model when chat template does not exist.

The best ICL performance is always achieved by *less* instruction-tuned models. As shown by Figure 1, for all 18 models, the peak of the curves is reached when $\lambda < 1$. It means scaling down instruction tuning strength to some degree enhances in-context learning ability. In addition, for 17 out of 18 models, except for Mixtral 8x22B, PAd improves ICL performance over both base and instruct models. For 15 out of 18 models, this improvement is greater than 0.5. The largest improvement we observe is 2.5 on Llama-3 8B. The best λ is oftentimes between 0.5 to 0.6. Similar trends are evident at the individual dataset level (Table 4).

The improvement on ICL is at the cost of losing some instruction following abilities as

[2]Note that both MC and FMC are standard evaluation protocols for multiple choice tasks used by LLM-foundry and MMLU.

[3]https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

[4]The GPT-4o version that we use is the May 2024 one.

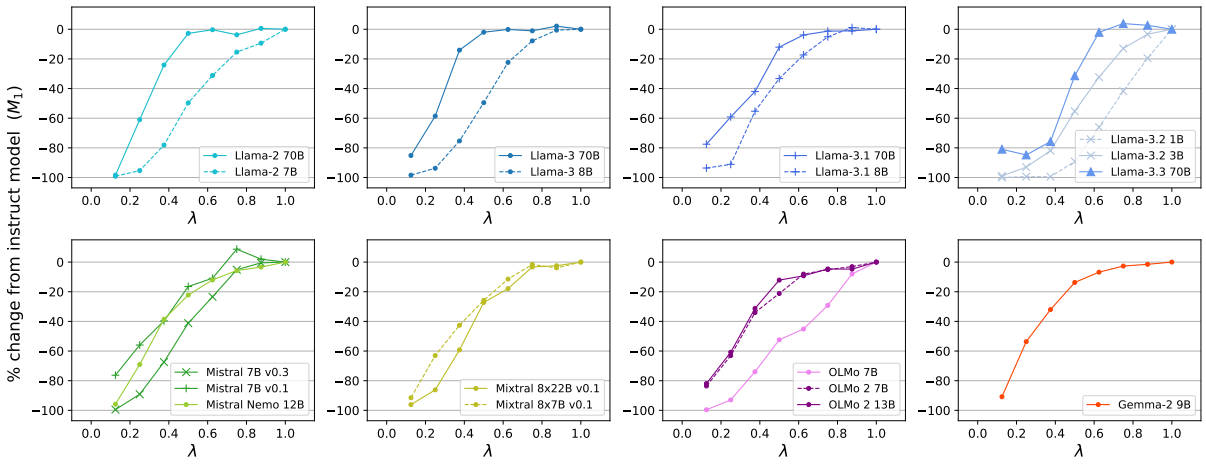

Figure 2: Performance on AlpacaEval 2.0: fractional difference (percent value) between the length controlled win rate of each partially adapted model M_λ and the instruct baseline M_1 against GPT-4 Preview (11/06).

measured by the AlpacaEval 2.0 win rate shown in Figure 2 and the last column of Table 1. In Table 1, $\delta_{\text{wr}} \equiv \text{wr}_{M_{\lambda^*}} - \text{wr}_{M_1}$ represents the *absolute* difference in win rate between the best PAd model for ICL (M_{λ^*}) and the instruct version (M_1). As shown in Figure 2, the best win rate is mostly achieved by the instruct model, except for a few cases where a marginally higher win rate is achieved when $0.6 < \lambda < 1$.

ICL can be improved with a small drop of instruction following abilities. We notice that for many models, especially the larger ones, the win-rate curve saturates to the instruct value for λ values well below 1. This implies that there are values of λ, in the range $\lambda^* \leq \lambda < 1$, where the AlpacaEval 2.0 performance does not drop significantly, yet there is still a gain on the ICL benchmark due to PAd. For instance, by allowing at most a 1% relative win rate decrease from the instruct model on AlpacaEval 2.0, we can get a $+5.9\%$ relative improvement on the ICL benchmark performance for Llama-2 70B ($\lambda = 0.625$), $+4.9\%$ for Llama-3 70B ($\lambda = 0.625$), $+1.7\%$ for Llama-3.3 70B ($\lambda = 0.75$).

5 Conclusion and Future Work

In this work, we study the performance trajectory between base and instruct models for 18 LLMs via the training-free partial adaptation method Fleshman and Van Durme (2024). We find that scaling down instruction tuning strength can benefit in-context learning tasks for all models across 21 datasets. However, this improvement is at the cost of losing instruction following ability.

Nonetheless, the observation that instruction following performance for larger models is not very sensitive to λ when $\lambda \lesssim 1$ suggests that scaling down instruction tuning strength to a small degree would consistently be beneficial. Hence, it would make sense to apply PAd at the end of post-training (e.g., replacing M_1 with M_{λ^*}) to further boost model performance. This might have already happened as Llama 3.3 (Meta, December 2024) used an annealing technique to average model checkpoints, and we also observed that PAd boosts Llama 2 ICL performance much more than Llama 3.3.

Future work can focus on better understanding why PAd improves ICL performance by studying its impact on each stage of supervised fine-tuning or RL. Another avenue of investigation is a thorough comparison of the training dynamics during instruction tuning with the model trajectory defined by varying λ in PAd. It has been suggested that the latter may indeed recapitulate the full training dynamics (Na et al., 2024).

Limitations

Our in-context learning benchmark is a collection of 21 common datasets spanning 6 broad types of tasks. The collection may however not be fully representative of the model ICL performance or its performance on other specific tasks. Similarly, we benchmark instruction following ability on AlpacaEval 2.0 (Dubois et al., 2024), which has a Spearman correlation of 0.98 with Chatbot Arena. However, it may not be fully representative of the model true instruction following performance. Further, we limit our study to models primarily trained

on English data and tasks in English, hence we leave testing the generalizability to other languages and multi-lingual models to future work.

References

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios N. Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: an open platform for evaluating llms by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, Brussels, Belgium. Association for Computational Linguistics.

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 preprint arXiv:1803.05457*.

Florin Cuconasu, Giovanni Trappolini, Nicola Tonellotto, and Fabrizio Silvestri. 2024. A tale of trust and accuracy: Base vs. instruct llms in rag systems. *arXiv preprint arXiv:2406.14972*.

Yang Deng, Wenqiang Lei, Wenxuan Zhang, Wai Lam, and Tat-Seng Chua. 2022. PACIFIC: Towards proactive conversational question answering over tabular and textual data in finance. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6970–6984, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv preprint arXiv:1903.00161*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

William Fleshman and Benjamin Van Durme. 2024. Readapt: Reverse engineered adaptation of large language models. *arXiv preprint arXiv:2405.15007*.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Raghavi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Emma Strubell, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Luke Zettlemoyer, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Olmo: Accelerating the science of language models.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Meta. December 2024. Llama-3.3 70b model card.

Meta. July 2024. Introducing llama 3.1.

Meta. September 2024. Llama 3.2.

mistral.ai. April 2024. mixtral-8x22b.

mistral.ai. July 2024. mistral-nemo.

Clara Na, Ian Magnusson, Ananya Harsh Jha, Tom Sherborne, Emma Strubell, Jesse Dodge, and Pradeep Dasigi. 2024. Scalable data ablation approximations for language models through modular training and merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21125–21141, Miami, Florida, USA. Association for Computational Linguistics.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Michal Guerquin, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James V. Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. 2 olmo 2 furious.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016a. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016b. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392.

Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

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

Raj Sanjay Shah, Kunal Chawla, Dheeraj Eidnani, Agam Shah, Wendi Du, Sudheer Chava, Natraj Raman, Charese Smiley, Jiaao Chen, and Diyi Yang. 2022. When flue meets flang: Benchmarks and large pre-trained language model for financial domain. *arXiv preprint arXiv:2211.00083*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

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. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

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*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2023. Large language models are not robust multiple choice selectors. *CoRR*, abs/2309.03882.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. 2021. TAT-QA: A question answering

benchmark on a hybrid of tabular and textual content in finance. 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 3277–3287, Online. Association for Computational Linguistics.

Capability	Domain	Dataset	Shots	Style	Size
World Knowledge	General	MMLU (Hendrycks et al., 2021)	5	MC, FMC	14042
		Trivia QA (Joshi et al., 2017)	1	G	1105
		Natural Questions (Kwiatkowski et al., 2019)	1	G	1032
Commonsense Reasoning	General	PIQA (Bisk et al., 2020)	1	MC	1838
		Winogrande (Sakaguchi et al., 2021)	1	MC	1267
		ARC Challenge (Clark et al., 2018)	1	MC	1172
		HellaSwag (Zellers et al., 2019)	1	MC	10042
Language Processing and Understanding	General	BBH (NLP) (Suzgun et al., 2023)	3	G, MC	3000
	Finance	FiQA (SA) (Shah et al., 2022)	5	MC, FMC	235
		FPB (SA) (Shah et al., 2022)	5	MC, FMC	970
		Headline (Shah et al., 2022)	5	MC, FMC	20547
		Flue (NER) (Shah et al., 2022)	20	G	98
Symbolic and Logical Problem Solving	General	BBH (Algo) (Suzgun et al., 2023)	3	G, MC	3000
		DROP (Dua et al., 2019)	1	G	1000
	Finance	TAT-QA (Zhu et al., 2021)	1	G	1668
		Pacific (Deng et al., 2022)	1	G	1982
Reading Comprehension	General	SQuAD (Rajpurkar et al., 2016a)	2	G	1000
		QuAC (Choi et al., 2018)	2	G	1000
	Finance	ConvFinQA (Chen et al., 2022)	1	G	5932
Retrieval-augmented Generation (RAG)	General	Natural Questions + Wiki (Kwiatkowski et al., 2019)	1	G	1105
		Trivia QA + Wiki (Joshi et al., 2017)	1	G	1032

Table 2: A complete list of the datasets composing our in-context few-shot learning evaluation benchmark. The last column (Size) shows the number of examples in each dataset.

A In-context Learning Benchmark Details

A.1 Datasets

Table 2 lists the datasets we used to build the ICL benchmark, which are organized in a taxonomy according to the ability they are supposed to test and the domain they are operating on.

- **World knowledge**: we include the widely used Massive Multitask Language Understanding (MMLU) benchmark (Hendrycks et al., 2021) and two open-domain QA tasks, Trivia QA (Joshi et al., 2017) and Natural Questions (Kwiatkowski et al., 2019).
- **Commonsense reasoning**: four datasets (PIQA (Bisk et al., 2020), Winogrande (Sakaguchi et al., 2021), ARC Challenge (Clark et al., 2018), and HellaSwag (Zellers et al., 2019)) to test different types of commonsense reasoning ability of the model.
- **Language processing and understanding**: we include five classic language processing or understanding tasks. BBH (NLP) are NLP tasks from Big Bench Hard (Suzgun et al., 2023), e.g., movie recommendation. FiQA (SA) and FFB (SA) are two sentiment analysis tasks, Headline is a headline classification tasl, Flue (NER) is a name entity recognition task, and all these four datasets are from FLUE (Financial Language Understanding Evaluation) benchmark (Shah et al., 2022).
- **Symbolic and logical problem solving**: BBH (Algo) contains algorithmic tasks (e.g., Boolean expressions) from Big Bench Hard (Suzgun et al., 2023). DROP (Dua et al., 2019) is a discrete reasoning QA dataset. TAT-QA (Zhu et al., 2021) and Pacific (Deng et al., 2022) are two financial table QA tasks.
- **Reading comprehension**: SQuAD (Rajpurkar et al., 2016a) and QuAC (Choi et al., 2018) are two general-domain reading comprehension QA datasets, and ConvFinQA (Chen et al., 2022) is a financial QA task.
- **Retrieval-augmented generation (RAG)**: we use questions from Natural Questions (Kwiatkowski et al., 2019) and Trivia QA (Joshi et al., 2017) to retrieve passages from Wikipedia, which creates two RAG evaluation tasks.

Template	Dataset	Style	Metric
mmlu_joint.j2	MMLU	FMC	Accuracy
mmlu_separate.j2	MMLU	MC	Accuracy
instruct_qa.j2	BBH	G	Accuracy
bbh_separate.j2	BBH	MC	Accuracy
sa_t4.j2	FBP (SA)	MC	Weighted F1
sa_t4_opt.j2	FBP (SA)	MC	Weighted F1
sa_t4_joint.j2	FBP (SA)	FMC	Weighted F1
ner_inline.j2	Flue (NER)	G	F1
simple_qa.j2	QuAC	G	String F1
	TAT-QA	G	Fin QA F1
	DROP	G	String F1
	ConvFinQA	G	Fin QA Accuracy
	SQuAD	G	String F1
	Natural Questions	G	String F1
	Trivia QA	G	String F1
simple_qa_new.j2	Natural Questions + Wiki	G	String F1
	Trivia QA + Wiki	G	String F1
simple_qa_mc.j2	ARC Challenge	MC	Accuracy
simple_qa_mc_opt.j2	Headline	MC	Average Weighted F1
simple_qa_mc_joint.j2	Headline	FMC	Average Weighted F1
asa_t4.j2	FiQA	MC	Weighted F1
asa_t4_opt.j2	FiQA	MC	Weighted F1
asa_t4_joint.j2	FiQA	FMC	Weighted F1
pacific.j2	Pacific	G	Fin QA F1
mc_concat.j2	HellaSwag	MC	Accuracy
	Winogrande	MC	Accuracy
	PIQA	MC	Accuracy

Table 3: Templates used for to evaluate each of the datasets. We also show the metrics used to evaluate the different datasets. If a single dataset is evaluated multiple times using different templates or styles, the final scores are aggregated by taking their maximum.

A.2 Evaluation Tasks

The in-context benchmark is composed of three categories of tasks.

- Multiple choice (MC): For multiple choice datasets, we use the model to score the likelihood of each of the possible choices $c \in \mathcal{C}$ and pick the highest ranking one, c^*,

$$c^* \equiv \underset{c \in \mathcal{C}}{\text{argmax}}\, \mathbb{P}(c \,|\, \text{prompt})/N(c) \tag{1}$$

  $N(c)$ is a possibly choice dependent normalization that we use to ameliorate possible biases of the model likelihood (Zheng et al., 2023). We consider 3 possibilities for N

$$N_{\text{base}}(c) = 1 \tag{2}$$
$$N_{\text{length}}(c) = |\texttt{tokens}(c)| \tag{3}$$
$$N_{\text{prior}}(c) = \mathbb{P}(\text{prefix}) \tag{4}$$

  where $\texttt{tokens}(c)$ is the list of tokens representing c and $\mathbb{P}(\text{prefix})$ is the probability that the model assigns to a generic prefix that does not depend on c, for instance the string "Answer: " (see Appendix A.3 for details). We calculate accuracy or F1 score for each of these choices of N and we aggregate the final results by taking the maximum across these scores.

- Fast multiple choice (FMC): Similar to MC, but instead of asking the model to score each possible response, the model is shown the possible choices as a bulleted list (in MMLU format (Hendrycks

et al., 2021)) and only the individual tokens corresponding to the bullets (A, B, C, ...) are scored and ranked

$$c^* \equiv \underset{c \in \{A, B, C, ...\}}{\text{argmax}} \mathbb{P}(c \,|\, \text{prompt}) \tag{5}$$

- Generation (G): The model generates a completion which is then parsed and compared to the ground truth answer. Evaluation metrics include string-F1 and Exact Match. The full list of evaluation metrics is shown in Table 3 and described in Appendix A.4.

A.3 Templates

In this section we report the templates that we use in our experiments. All of them are displayed in `jinja2` format.

In some of the templates below (`mmlu_separate.j2`, `bbh_separate.j2`, `sa_t4.j2`, `sa_t4_opt.j2`, `simple_qa_mc.j2`, `simple_qa_mc_opt.j2`, `asa_t4.j2`, `asa_t4_opt.j2`) the separator string `|||` appears. This is used to perform calibration following Eq. 4: the full template is obtained by replacing `|||` with the empty string, and the prefix appearing in Eq. 4 is obtained by splitting the prompt at `|||`:

```
_, prefix = template.split("|||")
template = template.replace("|||", "")
```

mmlu_joint.j2

```
{% set ENUM = ’ABCDEFGHIJKLM’ %}The following are multiple choice questions (with answers) about {{subject}}.
{% for example in examples %}
Question: {{ example.question }}
{% for choice in example.choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer: ({{ ENUM[example.gold] }}) {{ example.choices[example.gold] }}
{% endfor %}
Question: {{ question }}
{% for choice in choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer:
```

mmlu_separate.j2

```
The following are multiple choice questions (with answers) about {{subject}}.
{% for example in examples %}
Question: {{ example.question }}
Answer: {{ example.choices[example.gold] }}
{% endfor %}
Question: {{ question }}
|||Answer:
```

instruct_qa.j2

```
{% for example in examples %}{{ example.question }}
Answer: {{ example.gold }}

{% endfor %}{{ question }}
Answer:
```

bbh_separate.j2

```
{{instruction}}
{% for example in examples %}
Question: {{ example.question }}
Answer: {{ example.choices[example.gold] }}
{% endfor %}
Question: {{ question }}
|||Answer:
```

sa_t4.j2

```
{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment?
Answer: {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment?
|||Answer:
```

sa_t4_opt.j2

```
{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment?
Options:
{% for choice in ex.choices %}- {{ choice }}
{% endfor %}Answer: {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment?
Options:
{% for choice in choices %}- {{ choice }}
{% endfor %}|||Answer:
```

sa_t4_joint.j2

```
{% set ENUM = 'ABCDEFGHIJKLM' %}{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment?
{% for choice in ex.choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer: ({{ ENUM[ex.gold] }}) {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment?
{% for choice in choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer:
```

simple_qa.j2

```
{% for example in examples %}{{ example.question }}
Answer: {{ example.gold }}

{% endfor %}{{ question }}
Answer:
```

simple_qa_new.j2

```
{% for example in examples %}{{ example.sources|join('\n\n') }}

{{ example.question }}
Answer: {{ example.gold }}

{% endfor %}{{ sources|join('\n\n') }}

{{ question }}
Answer:
```

simple_qa_mc.j2

```
{% for example in examples %}{{ example.question }}
Answer: {{ example.choices[example.gold] }}

{% endfor %}{{ question }}
|||Answer:
```

simple_qa_mc_opt.j2

```
{% for ex in examples %}{{ ex.question }}
Options:
{% for choice in ex.choices %}- {{ choice }}
{% endfor %}Answer: {{ ex.choices[ex.gold] }}

{% endfor %}{{ question }}
Options:
{% for choice in choices %}- {{ choice }}
{% endfor %}|||Answer:
```

simple_qa_mc_joint.j2

```
{% set ENUM = ’ABCDEFGHIJKLM’ %}{% for ex in examples %}{{ ex.question }}
{% for choice in ex.choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer: ({{ ENUM[ex.gold] }}) {{ ex.choices[ex.gold] }}

{% endfor %}{{ question }}
{% for choice in choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer:
```

asa_t4.j2

```
{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment on {{ ex.target }}?
Answer: {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment on {{ target }}?
|||Answer:
```

asa_t4_opt.j2

```
{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment on {{ ex.target }}?
Options:
{% for choice in ex.choices %}- {{ choice }}
{% endfor %}Answer: {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment on {{ target }}?
Options:
{% for choice in choices %}- {{ choice }}
{% endfor %}|||Answer:
```

asa_t4_joint.j2

```
{% set ENUM = ’ABCDEFGHIJKLM’ %}{% for ex in examples %}{{ ex.sentence }}
Question: what is the sentiment on {{ ex.target }}?
{% for choice in ex.choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer: ({{ ENUM[ex.gold] }}) {{ ex.choices[ex.gold] }}

{% endfor %}{{ sentence }}
Question: what is the sentiment on {{ target }}?
{% for choice in choices %}({{ ENUM[loop.index0] }}) {{ choice }}
{% endfor %}Answer:
```

pacific.j2

```
{% for example in examples %}{{ example.question }}
{{ example.gold }}

{% endfor %}{{ question }}
```

mc_concat.j2

```
{% for example in examples %}{{ example.question }}{{ example.choices[example.gold]
    }}

{% endfor %}{{ question }}
```

A.4 Metrics

Table 3 lists the metrics used to evaluate each dataset in our benchmark.

- **Accuracy**: For classification tasks, it checks whether the predicted label matches the gold label. For generation tasks, it checks whether the generated answer matches the gold answer.
- **Weighted F1**: Calculate F1 scores for each class, and find their average weighted by support (the number of true instances for each class).

- **F1**: This metric is only used for the Flue (NER) task. For each entity type, there are a list of gold entities and a list of model-generated entities. True positive is the number of overleaped between ground-truth and model generations. False positive is the number of entities that the model generates but are not ground-truth. False negative is the number of entities that are gold but the model does not generate.

- **String F1**: We use the same evaluation script from SQuAD (Rajpurkar et al., 2016b), in which gold and generated answers are treated as two bags of words. String F1 is the F1 score between these two bags of words are computed.

- **Fin QA F1**: This metric is the same as String F1, except for two cases. When the gold answer is a *number*, we extract and convert the model generation to a number and check if it matches the gold number. When the gold answer is yes or no, we check if the first word of model generation matches the gold answer.

- **Fin QA Accuracy**: This metric is similar to Fin QA F1, except that we replace String F1 with String EM (Exact Match) because the answers are mostly short.

- **Average Weighted F1**: This metric is used when there are multiple groups of multi-choice classification tasks. We compute the weighted F1 within each group and then take the average across groups.

All metric sores are in a scale of 0 to 100. Therefore, we are able to average dataset-level scores into one single model-level score.

B Additional Results

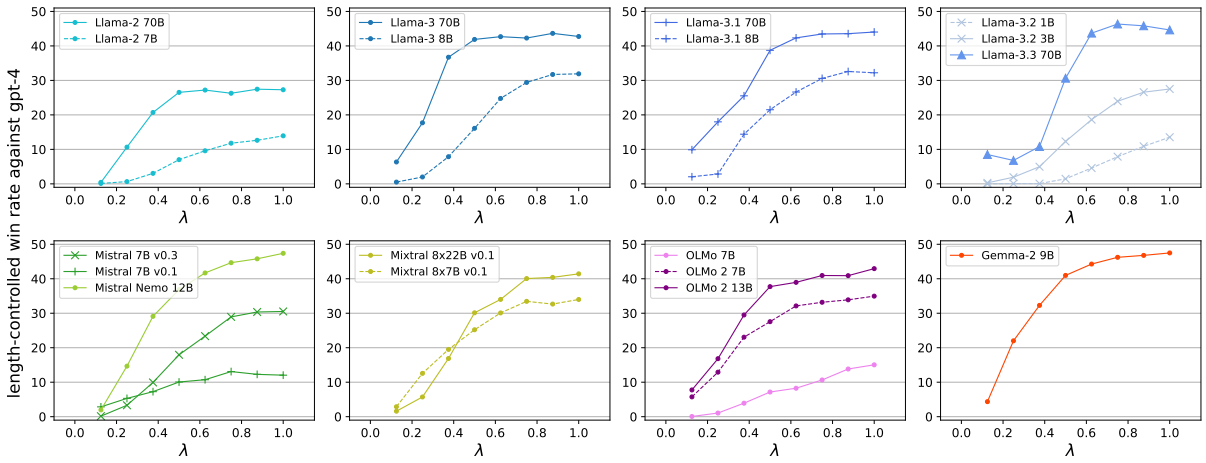

Figure 3: Performance on AlpacaEval 2.0: the length controlled win rates of each partially adapted model M_λ against GPT-4 Preview (11/06).

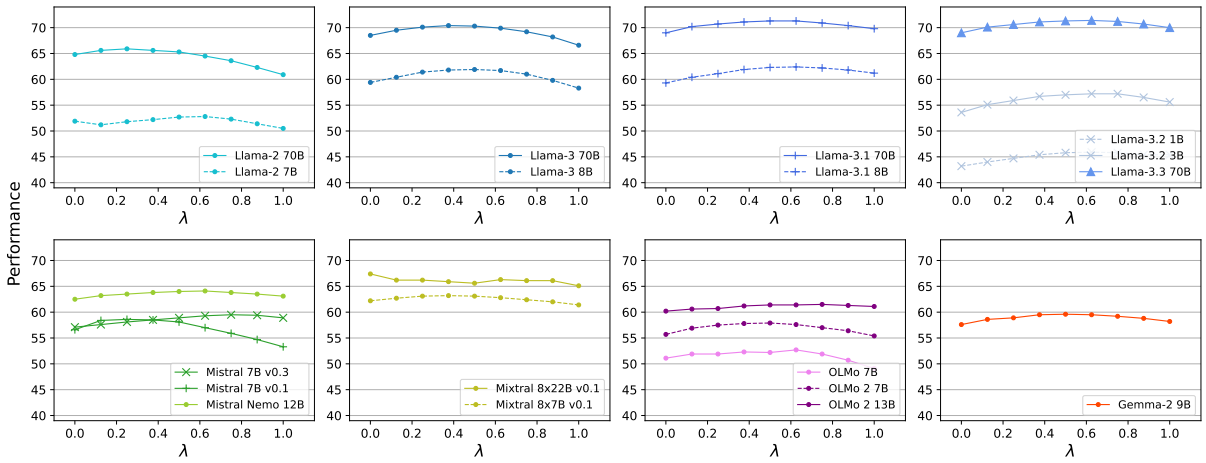

Figure 4: Performance on the in-context learning benchmark: absolute performance of each partially adapted model M_λ for all the models we have tested.

Model	MMLU (base)	MMLU (inst.)	MMLU	ARC Challenge	BBH (Algo)	BBH (NLP)	ConvFinQA	DROP	Headline	Flue (NER)	FiQA (SA)	HellaSwag	Natural Questions	Pacific	PIQA	QuAC	Natural Questions + Wiki	Trivia QA + Wiki	SQuAD	TAT-QA	Trivia QA	Winogrande
Llama-3 70B	79.1	80.6	$81.0^{4/8}$	$74.0^{7/8}$	$57.9^{5/8}$	$78.7^{2/8}$	$69.9^{6/8}$	$81.5^{3/8}$	$91.6^{1/8}$	65.6^{1}	$83.5^{6/8}$	$85.6^{1/8}$	$39.8^{3/8}$	$67.0^{4/8}$	$83.8^{2/8}$	$60.3^{3/8}$	$35.9^{5/8}$	80.6^{0}	$53.2^{3/8}$	$64.5^{2/8}$	$82.0^{4/8}$	$79.9^{2/8}$
Llama-3 8B	65.2	66.7	$66.9^{7/8}$	$64.8^{5/8}$	$42.9^{7/8}$	$66.8^{6/8}$	$60.1^{5/8}$	$64.0^{3/8}$	$90.1^{4/8}$	$67.0^{6/8}$	$79.4^{7/8}$	$79.4^{2/8}$	$31.5^{2/8}$	$55.0^{6/8}$	$80.2^{4/8}$	$52.0^{5/8}$	$35.5^{3/8}$	$74.3^{2/8}$	$48.5^{7/8}$	$52.4^{3/8}$	$68.1^{5/8}$	$73.0^{1/8}$
Llama-3.1 70B	78.8	82.7	82.7^{1}	$72.1^{5/8}$	$60.8^{7/8}$	$80.5^{4/8}$	$69.7^{3/8}$	$82.3^{3/8}$	$91.8^{2/8}$	$65.6^{5/8}$	$83.3^{5/8}$	$85.8^{2/8}$	$42.6^{4/8}$	$67.1^{3/8}$	$83.9^{3/8}$	$59.4^{3/8}$	$40.1^{7/8}$	$81.9^{1/8}$	$53.5^{4/8}$	$67.1^{4/8}$	$83.2^{1/8}$	$81.0^{1/8}$
Llama-3.1 8B	65.6	68.5	$68.5^{7/8}$	$61.9^{6/8}$	$46.9^{7/8}$	$66.7^{5/8}$	$59.9^{5/8}$	$63.5^{4/8}$	$89.6^{6/8}$	$67.5^{5/8}$	78.6^{1}	$79.5^{2/8}$	$32.8^{4/8}$	55.3^{1}	$80.1^{3/8}$	$52.4^{5/8}$	$39.8^{7/8}$	$74.5^{1/8}$	$44.5^{7/8}$	$54.0^{6/8}$	$68.1^{4/8}$	$72.4^{2/8}$
Llama-3.2 3B	56.2	61.2	$61.6^{6/8}$	$55.6^{5/8}$	$43.6^{6/8}$	$62.3^{6/8}$	$53.8^{4/8}$	$53.0^{5/8}$	$89.7^{5/8}$	$60.7^{4/8}$	$78.3^{7/8}$	$74.0^{2/8}$	$28.9^{3/8}$	$48.3^{7/8}$	$77.8^{3/8}$	48.9^{1}	$37.9^{3/8}$	$68.9^{3/8}$	44.8^{1}	$49.6^{5/8}$	$57.1^{3/8}$	$67.2^{1/8}$
Llama-3.2 1B	37.7	45.2	$45.2^{7/8}$	$44.5^{5/8}$	$29.2^{6/8}$	$53.3^{3/8}$	$38.1^{5/8}$	$30.9^{3/8}$	$83.5^{5/8}$	58.3^{1}	72.3^{1}	$63.0^{2/8}$	$18.8^{4/8}$	$30.4^{7/8}$	$76.0^{2/8}$	$40.0^{5/8}$	$31.6^{1/8}$	$54.4^{3/8}$	33.5^{1}	35.1^{1}	$37.2^{4/8}$	60.0^{0}
Llama-3.3 70B	78.9	82.7	82.7^{1}	$73.1^{5/8}$	61.1^{1}	$80.5^{4/8}$	$69.5^{4/8}$	$82.2^{4/8}$	$92.0^{4/8}$	$66.7^{6/8}$	$83.2^{7/8}$	$85.7^{2/8}$	$40.6^{4/8}$	$67.4^{5/8}$	$84.2^{4/8}$	$58.9^{5/8}$	$38.1^{7/8}$	$81.7^{2/8}$	$59.4^{5/8}$	$66.4^{3/8}$	$83.4^{2/8}$	$80.8^{1/8}$
Llama-2 70B	69.2	63.6	$70.2^{2/8}$	$67.7^{2/8}$	$53.3^{2/8}$	$71.7^{2/8}$	$61.7^{4/8}$	$71.6^{2/8}$	$91.2^{2/8}$	62.7^{1}	$81.8^{6/8}$	$83.4^{2/8}$	$42.8^{1/8}$	$56.3^{2/8}$	$82.8^{3/8}$	$54.7^{4/8}$	39.5^{0}	$79.5^{1/8}$	$50.6^{6/8}$	$55.9^{4/8}$	80.3^{0}	$79.6^{1/8}$
Llama-2 7B	44.1	47.3	$47.9^{6/8}$	$55.2^{3/8}$	33.6^{0}	$57.0^{6/8}$	$43.2^{5/8}$	$42.2^{5/8}$	$86.9^{7/8}$	$64.6^{2/8}$	$74.1^{6/8}$	$75.9^{3/8}$	28.5^{0}	$38.7^{7/8}$	$78.8^{4/8}$	$43.8^{5/8}$	33.6^{0}	67.4^{0}	38.8^{0}	$39.6^{5/8}$	$59.7^{2/8}$	$67.9^{4/8}$
Gemma-2 9B	72.0	72.6	$73.6^{3/8}$	$71.6^{7/8}$	$48.5^{3/8}$	$72.8^{5/8}$	$52.6^{5/8}$	$54.7^{4/8}$	$90.4^{2/8}$	$45.7^{6/8}$	82.4^{1}	$80.7^{4/8}$	$34.7^{3/8}$	$39.9^{3/8}$	$82.1^{4/8}$	$33.3^{6/8}$	$42.7^{1/8}$	$76.1^{3/8}$	$41.2^{7/8}$	41.0^{1}	$67.4^{2/8}$	$75.5^{1/8}$
Mixtral 8x22B v0.1	76.2	76.6	$76.7^{6/8}$	71.5^{1}	55.6^{0}	74.7^{0}	$72.6^{6/8}$	$78.4^{4/8}$	89.8^{0}	66.7^{0}	82.9^{1}	$82.9^{7/8}$	$42.1^{2/8}$	64.3^{0}	$75.5^{5/8}$	$56.1^{6/8}$	$42.1^{2/8}$	$82.0^{2/8}$	$49.5^{7/8}$	$64.0^{5/8}$	82.4^{0}	$68.7^{7/8}$
Mixtral 8x7B v0.1	69.1	69.3	$70.0^{3/8}$	70.8^{1}	$48.6^{3/8}$	$71.5^{5/8}$	$64.0^{7/8}$	$67.9^{3/8}$	$89.2^{6/8}$	$64.3^{7/8}$	$79.2^{7/8}$	81.8^{1}	37.9^{0}	$56.2^{2/8}$	$76.6^{7/8}$	$50.6^{6/8}$	$40.5^{2/8}$	78.5^{0}	$44.3^{3/8}$	$56.7^{3/8}$	$77.5^{4/8}$	65.1^{0}
Mistral 7B v0.1	56.9	49.9	$57.5^{2/8}$	$64.1^{2/8}$	$45.5^{2/8}$	$65.0^{2/8}$	$54.5^{2/8}$	$58.8^{4/8}$	$90.5^{3/8}$	61.5^{0}	$78.2^{6/8}$	$77.0^{1/8}$	$33.1^{1/8}$	$46.6^{1/8}$	$74.4^{2/8}$	$51.6^{7/8}$	$37.1^{5/8}$	$69.8^{2/8}$	$47.4^{7/8}$	$50.4^{2/8}$	$65.8^{1/8}$	$62.9^{2/8}$
Mistral 7B v0.3	59.7	59.9	$60.0^{2/8}$	$66.0^{7/8}$	$44.3^{3/8}$	$65.5^{5/8}$	$56.0^{6/8}$	$55.6^{5/8}$	90.4^{1}	$64.7^{5/8}$	80.3^{1}	79.5^{1}	$28.7^{4/8}$	$48.6^{4/8}$	75.7^{1}	$54.9^{7/8}$	$36.1^{4/8}$	$71.8^{6/8}$	44.3^{1}	$49.4^{6/8}$	$66.0^{6/8}$	$64.2^{6/8}$
Mistral Nemo 2407	68.4	69.1	$69.4^{4/8}$	$63.2^{5/8}$	$46.5^{5/8}$	$70.6^{5/8}$	$64.1^{6/8}$	$72.1^{4/8}$	88.1^{0}	64.4^{1}	80.4^{1}	$81.6^{4/8}$	$32.2^{3/8}$	56.4^{1}	$81.8^{5/8}$	57.9^{1}	$38.6^{3/8}$	$74.6^{1/8}$	$49.1^{5/8}$	$58.1^{5/8}$	72.1^{0}	$75.7^{4/8}$
OLMo 7B 0724	52.0	53.0	$54.4^{4/8}$	$51.4^{6/8}$	$35.9^{3/8}$	57.4^{0}	$35.8^{5/8}$	$53.1^{4/8}$	$88.7^{2/8}$	63.4^{0}	$74.4^{5/8}$	79.5^{1}	33.0^{0}	$36.8^{5/8}$	$79.9^{3/8}$	$32.0^{5/8}$	$35.2^{3/8}$	64.3^{0}	$41.0^{1/8}$	$40.9^{5/8}$	53.7^{0}	$68.6^{2/8}$
OLMo 2 13B 1124	67.0	66.4	67.0^{0}	67.3^{1}	39.0^{1}	66.5^{1}	$50.3^{6/8}$	73.4^{0}	87.0^{0}	$63.5^{7/8}$	77.4^{1}	85.9^{1}	39.8^{0}	$51.6^{5/8}$	81.9^{1}	$48.5^{6/8}$	42.8^{0}	71.7^{0}	43.7^{1}	$51.3^{6/8}$	68.2^{0}	$75.7^{1/8}$
OLMo 2 7B 1124	62.7	61.2	$63.1^{3/8}$	$64.2^{6/8}$	$33.9^{6/8}$	$63.0^{7/8}$	$44.3^{6/8}$	$63.2^{1/8}$	$85.8^{4/8}$	59.5^{0}	74.7^{0}	83.6^{1}	$32.5^{3/8}$	$44.3^{4/8}$	$80.7^{7/8}$	$46.9^{4/8}$	$41.2^{1/8}$	$67.4^{2/8}$	$43.6^{3/8}$	$46.2^{2/8}$	$64.5^{1/8}$	$71.3^{3/8}$

Table 4: Fine-grained results for each model and each dataset in our benchmark. Each entry reports the best score achieved and the value of λ at which such score was achieved in the format score$^{\lambda}$. We also report MMLU performances for the base and instruct version of every model in the first two columns.