Keep the Alignment, Skip the Overhead: Lightweight Instruction Alignment for Continually Trained LLMs

Ishan Jindal [1] **Chandana Badrinath** [1] **Lakkidi Vinay** [1] **Pranjal Bharti** [1] **Sachin Dev Sharma** [1]

Abstract

Instruction fine-tuning aligns language models with human intent but is computationally costly. Continuous pretraining on domain-specific data, while effective for adaptation, can degrade instruction-following capabilities. We introduce **instruction residuals**—the parameter delta between an instruction-tuned model and its base model—as a lightweight mechanism to recover instruction alignment post adaptation. Instruction residuals can be transferred across checkpoints within the same model family, enabling restoration of instruction-following behavior without full retraining. We evaluate our method on LLaMa and Qwen models under domain shifts of up to 1B tokens, showing that instruction residuals effectively preserve alignment while allowing continual domain learning. Our results establish a practical framework for modular, compute-efficient instruction retention in evolving language models.

1. Introduction

Recently, autoregressive Large Language Models (LLMs) have demonstrated remarkable progress across a wide range of natural language tasks, including understanding, reasoning, and coding (Achiam et al., 2023; Team et al., 2024; Touvron et al., 2023; Roziere et al., 2023; Yang et al., 2024a). These models are pre-trained using a causal language modeling objective, producing a *Base model* that excels at linguistic coherence but lacks alignment with human preferences (Ouyang et al., 2022). To bridge this gap, *Instruction fine-tuning* is applied, refining models to better follow human instructions (Rafailov et al., 2024; Ethayarajh et al., 2024), resulting in *Instruction models*.

Instruction fine-tuning is costly, often requiring tens of millions of human-annotated examples (e.g., 10M for LLaMa 3 Instruct(AI@Meta, 2024)), and complex optimization techniques such as RLHF, PPO (Ouyang et al., 2022), and DPO (Rafailov et al., 2024). Simultaneously, keeping LLMs updated with new information necessitates *continuous pre-training*, where the base model is further trained on newly collected data (Gao et al., 2020; Tokpanov et al., 2024; Ibrahim et al., 2024). For instance, the LLaMa 3.1 base model incorporates more high-quality data over LLaMa 3 (Dubey et al., 2024), and Qwen 2.5 improves upon Qwen 2 (Team, 2024).

A major challenge arises when integrating new knowledge while retaining instruction-following abilities. Continuous pre-training may overwrite instruction-tuned parameters, leading to a loss of instruction-following capabilities. While prior research has explored mitigating catastrophic forgetting in base models (Xie et al., 2023; Ibrahim et al., 2024), little attention has been given to its effects on instruction models. These observations motivate three critical research questions:; **Q1**: *How does continuous pre-training affect the instruction-following capabilities of an instruction-tuned model?* **Q2**: *If instruction capabilities degrade, how can they be recovered efficiently?* **Q3**: *Is full-scale instruction fine-tuning necessary after updating a model's knowledge?*

We empirically investigate these questions by exploring an alternative to traditional instruction fine-tuning: **instruction residuals**. Instead of re-running expensive fine-tuning procedures, we propose a simple yet effective method—*extracting instruction residuals* from an existing instruction-tuned model and applying them to a newly pre-trained base model. This approach allows us to recover instruction-following abilities while preserving newly learned knowledge. Our key findings are:

- Continuous pre-training of an instruction model significantly reduces its instruction-following capabilities and should be avoided. Section 4.1.
- Continuous pre-training the base model first, followed by instruction tuning, preserves both knowledge and instruction capabilities. Section 4.4.
- Instruction capabilities can be transferred across models *within the same base model lineage* using instruc-

*Equal contribution [1]Samsung R&D Institute India - Delhi. Correspondence to: Ishan Jindal <ishan.jindal@samsung.com>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

Table 1: Key Notations

Symbol	Meaning
θ_i	Instruction-tuned LLM Params
θ_b	Pretrained base LLM Params
θ_i^x, θ_b^y	Instruction/base model at data x, y
Θ_r^z	Instruction residual: $\theta_i^x - \theta_b^y$
$\theta_b^x + \Theta_r^z$	Reconstructed instruction model

tion residuals, computed as the difference between the instruction-tuned and base models. Section 4.3.

- Traditional instruction fine-tuning is not necessary for continuously pre-trained base models—instruction capabilities can be restored via instruction residuals. Section 4.2.
- Instruction residuals outperform both continuously pre-trained and instruction-degraded models, recovering capabilities with minimal performance trade-offs.

To our knowledge, this is the first systematic study analyzing the portability of instruction capabilities across models derived from the same base model. We validate our findings on LLaMa 3, LLaMa 3.1, Qwen 2, and Qwen 2.5, conducting extensive experiments across varying pre-training data sizes and model configurations (Section 3).

2. Background

In this investigation, we consider LLM families where both the base and instruction-tuned models are publicly available. Let $\theta_b^{d_1}$ denote parameters of a base autoregressive model trained on dataset d_1, and $\theta_i^{d_1 v_1}$ the parameters after instruction tuning on dataset v_1. Given a new pretraining dataset d_2, our goal is to obtain a d_2-specific LLM satisfying:

P1 Since d_2 is small (e.g., <1B tokens), we want the model to retain the language understanding acquired from d_1, as d_2 alone is insufficient to train even moderate-scale LLMs (e.g., 7B).

P2 The model should match the instruction-following capabilities of the instruction-tuned model.

We consider two possible strategies for adapting LLMs to a new domain (d_2) while retaining instruction capabilities:

Base-CP-Instruct: Start with the base model $\theta_b^{d_1}$, apply continual pretraining (CP) on d_2 to obtain $\theta_b^{d_1 d_2}$ (preserving P1 via techniques like Ibrahim et al. (2024)), and then instruction-tune on v_1 to get $\theta_i^{d_1 d_2 v_1}$ (gain P2). See Figure 1(a). This method retains prior knowledge while enabling domain adaptation but requires costly and often unstable instruction tuning, compounded by the lack of publicly available instruction datasets (Sun et al., 2024; Wang et al., 2022; Xu et al., 2023).

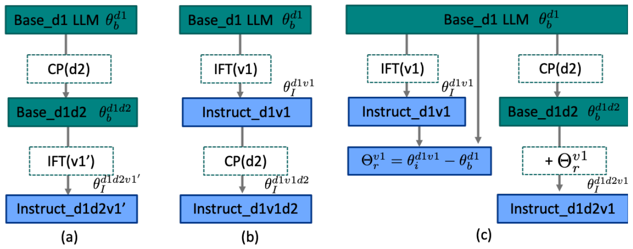

Figure 1: Overview of possible combinations for continuous pre-training (CP) and instruction fine-tuning (IFT), a) CP(d2) is applied on base LLM first and then IFT(v1) is applied; b) CP(d2) is applied on instruct LLM; c) CP(d2) is applied on base LLM and then Θ is added. Processes are denoted as dotted squares. GREEN denotes pre-trained only LLM and BLUE denotes instruction-tuned LLM.

Base-Instruct-CP: Begin with the instruction-tuned model $\theta_i^{d_1 v_1}$ and directly continue pretraining on d_2, aiming to retain both P1 and P2. See Figure 1(b). While this setting is less expensive and bypasses the need for instruction tuning, our experiments found no evidence supporting its effectiveness—rather, we observed degradation in instruction-following ability, resulting in $\theta_i^{d_1 v_1 d_2}$.

2.1. Instruction Residuals

To avoid expensive re-tuning, we explore a lightweight alternative: **instruction residuals**—parameter deltas between instruction-tuned and base models. We hypothesize that these residuals can be transferred to updated base models to regain instruction-following behavior efficiently. As shown in Figure 1 (c), we compute the instruction residual between a instruction following LLM $\theta_i^{d_1 v_1}$ and its corresponding base model $\theta_b^{d_1}$ in the parametric space as

$$\Theta_r^{v_1} = \theta_i^{d_1 v_1} - \theta_b^{d_1}. \tag{1}$$

This residual computation is inspired by the parameter efficient fine-tuning of LLMs such as low-rank adaptation (LoRA (Hu et al.), QLoRA(Dettmers et al., 2024), DoRA (Liu et al., 2024) etc). In these techniques instead of fine-tuning a large weight matrix W for a given layer, a low-rank ΔW matrix is learned, which contains the new information to be integrated with the original model that is $W_{updated} = W + \Delta W$. These techniques add new information/capabilities to the original model often with fewer parameters defined by *rank* of ΔW. With the full ΔW rank, it is similar to fine-tuning the whole model (Hu et al.).

Inspired by this idea of weight addition to learning a new capability, we first extract the instruction capability by subtracting the base LLM weights from its corresponding instruction-tuned LLM weights as in 1, termed as *instruction residuals*, and add this instruction residual to the continu-

Table 2: Impact of continual pretraining on LLaMa 3 Base (`L3b`) and the LLaMa 3 instruction tuned (`L3i`) models w.r.t varying number of new tokens. Also, depicts the usefulness of the instruction residual technique to regain instructional capabilities. Refer to Appendix A for clear notation meaning.

Benchmark	Metric	L3b				L3i				L3b + 3Lr		
# of new tokens →		org.	+100M	+500M	+1B	org.	+100M	+500M	+1B	100M	500M	1B
IFEval	ILL_acc	19.06	17.75	19.30	20.14	53.36	45.68	45.32	41.01	57.67	56.47	57.79
	ILS_acc	17.87	16.31	17.87	17.87	47.84	40.41	38.61	35.25	51.68	51.44	51.68
	PLL_acc	09.98	09.61	10.54	11.09	41.77	34.20	33.64	28.10	44.18	42.88	44.36
	PLS_acc	09.06	08.69	09.61	09.80	35.30	29.21	26.80	22.55	37.52	36.78	37.52
MMLU	acc	62.14	63.76	63.77	63.62	63.83	66.04	65.38	65.52	67.69	67.16	67.51
MMLU-Pro	EM	34.51	34.92	34.70	35.49	39.70	36.39	35.76	35.52	40.72	40.84	40.27
GSM8K	EM	49.58	48.14	47.54	47.08	75.06	69.22	68.08	68.01	74.83	73.92	73.16
	strict-EM	49.20	34.04	36.62	39.04	74.98	65.35	59.74	55.42	46.93	55.27	49.51
Sub-Average		31.43	29.15	29.99	30.52	**53.98**	48.31	46.67	43.92	52.65	53.10	52.73
Winogrande	acc	73.16	72.14	71.59	71.27	71.74	72.22	71.74	71.74	71.43	72.06	71.19
Hellaswag	acc	60.12	60.17	60.27	60.23	57.70	58.69	58.73	59.00	59.30	59.16	59.39
	acc_n	79.22	78.62	78.20	78.36	75.76	77.84	77.68	77.80	79.01	79.14	79.00
ARC_easy	acc	80.39	81.14	80.60	80.60	81.52	79.71	79.63	79.46	82.20	82.32	82.03
	acc_n	77.78	80.30	79.67	79.38	79.63	77.53	77.65	77.44	79.00	79.25	79.08
Piqa	acc	79.54	80.09	80.30	80.20	78.56	79.87	79.49	79.49	80.25	80.20	80.41
	acc_n	80.74	81.56	81.34	80.96	78.62	80.41	79.98	79.98	80.63	80.74	80.90
Sub-Average		75.85	**76.29**	76.00	75.86	74.79	75.18	74.99	74.99	75.97	76.12	76.00
T_mc2	acc	43.94	47.64	47.29	47.41	51.67	51.80	51.55	51.68	56.13	55.73	**56.17**
Average		51.64	50.93	51.20	51.41	62.94	60.28	59.36	58.00	63.07	63.34	63.12

ously pre-trained base LLM on new skill d_2 i.e.

$$\theta_i^{d_1 d_2 v_1} = \theta_b^{d_1 d_2} \oplus \Theta_r^{v_1}, \tag{2}$$

where $\oplus$ represents element-wise addition. These tensor addition and subtraction to regain the instruction capabilities do not incur heavy computation costs making the instruction-tuned LLM readily available once the new knowledge is learned by the base LLM. One major limitation of this work is that if the base LLM and its corresponding instruction-tuned LLM are not available then the instruction residuals from 1 won't be available and hence requires a full cycle of instruction fine-tuning to regain this ability.

3. Experiment Settings

We conduct experiments on two popular LLM families: LLaMa (v3, v3.1) and Qwen (v2, v2.5). For all experiments, we select only those models where both the base and corresponding instruction-tuned variants are publicly available. We use a carefully curated news dataset from Dec 2023 to Sep 2024 for continual pretraining, ensuring no data contamination. Evaluation is performed using 9 standard benchmarks for instruction following (4), knowledge and reasoning (4), and truthfulness (1), using EleutherAI's evaluation harness (Gao et al., 2021). Full details on datasets, architectures, and preprocessing are provided in Appendix B.

4. Results and Analysis

We evaluate how continual pretraining affects the instruction-following abilities of both base (`L3b`) and instruction-tuned (`L3i`) LLaMa 3 models.

4.1. Impact of Continual Pretraining

Continual pretraining on newly collected data (100M–1B tokens) significantly degrades instruction performance in `L3i`. As shown in Table 2, `L3i` loses up to 10 points on instruction tasks (53.8 to 43.92 sub-average), with an average drop of 3 points. This decline grows with more data due to the shift in learned representations, disrupting earlier instruction alignment. In contrast, `L3b` remains stable across pretraining sizes, confirming that base models do not suffer catastrophic forgetting in instruction tasks.

Key takeaway: Updating model knowledge via continual pretraining degrades instruction performance, particularly in instruction-tuned models, and requires a correction.

4.2. Restoring Instructional Abilities

We apply instruction residuals—the parameter delta between the base and instruction-tuned model—to recover instruction-following performance in pretrained models. As seen in the last block of Table 2, this technique (`L3b + 3Lr`) not only restores but improves performance, surpassing the original `L3i` model by up to 5 points on 1B tokens (43.92 to

52.73 sub-average). This gain is consistent across different sub-averages.

Key takeaway: Instruction residuals offer a lightweight and effective alternative to full re-tuning, efficiently restoring instruction alignment after continual pretraining.

4.3. Instruction Portability across LLM Families

We investigate the portability of instruction tuning across models using instruction residuals—defined as the parameter delta between instruction-tuned and base models. Specifically, we evaluate whether applying these residuals to compatible base models improves instruction-following performance. For instance, applying residuals from LLaMa 3.1's instruction-tuned variant to LLaMa 3's base model (`+3.1Lr`) boosts its average score from 51.64 to 64.25, even outperforming the directly fine-tuned LLaMa 3 model (`L3i`) (see Table 4, Appendix A).

In the LLaMa family, residuals from the more capable LLaMa 3.1 instruction-tuned model consistently uplift both LLaMa 3 and LLaMa 3.1 bases. Similarly, within the Qwen family, applying Qwen 2.5 residuals to Qwen 2 leads to notable gains. Importantly, in all cases, models with residuals outperform their original base variants—highlighting that instruction capabilities are portable within the same model family. However, cross-family transfer fails: applying residuals across different architectures (e.g., Qwen to LLaMa) yields suboptimal or degraded performance. This behavior mirrors LoRA's architecture-specific nature—residuals encode family-specific inductive biases that do not generalize across tokenization schemes or pretraining corpora. Overall, instruction residuals offer a low-cost, modular method to improve instruction-following without full fine-tuning—but only within the bounds of a shared architecture.

4.4. Instruction Residual Applicability to Derived LLMs

We evaluate the portability of instruction residuals to LLMs derived from a common base model. Specifically, we apply LLaMa 3 and 3.1 residuals to *cerebras/Llama3-DocChat-1.0-8B*, a model fine-tuned on the ChatQA dataset for document-based QA. As shown in Table 3, both residuals enhance instruction-following ability, with LLaMa 3.1 residuals providing up to a 6-point improvement. This demonstrates that **instruction residuals are transferable across models derived from the same base**, even when task specialization is introduced.

Table 3: Applicability of instruction residual approach on publicly available *Llama3-DocChat-1.0-8B* (DocChat) LLM that was built on top of LLaMa 3 base model. Here, +3Lr and +3.1Lr are the instruction residuals from LLaMa 3 and LLaMa 3.1, respectively integrated to the original DocChat LLM.

Benchmark	**Metric**	DocChat	+3Lr	+3.1Lr
IFEval	ILL_acc	38.25	49.64	56.71
	ILS_acc	34.65	46.04	53.12
	PLL_acc	24.95	37.34	44.36
	PLS_acc	20.89	32.90	39.74
MMLU	acc	62.96	62.31	65.35
MMLU-Pro	EM	36.36	39.66	39.36
GSM8K	EM	57.09	78.54	74.53
	strict-EM	56.94	77.48	69.90
Winogrande	acc	74.27	71.27	73.32
Hellaswag	acc	61.68	57.57	59.51
	acc_n	80.36	75.79	78.39
ARC_easy	acc	82.11	81.02	80.22
	acc_n	81.52	79.00	77.99
Piqa	acc	80.47	78.13	78.78
	acc_n	81.61	77.97	78.62
T_mc2	acc	45.35	49.54	50.82
Average		57.47	62.14	**63.80**

5. Related Work

Continual Learning: Continual or continuous pre-training enables LLMs to adapt to new data without forgetting prior knowledge (Caccia et al., 2020; Le Scao et al., 2023; Ibrahim et al., 2024). This approach has been widely used to acquire domain-specific skills and languages (Yadav et al., 2023; Ma et al., 2023; Yang et al., 2024b; Gogoulou et al., 2023). For instance, Yang et al. (2024b) combine continual pre-training with instruction tuning to specialize models, though the mechanisms behind its effectiveness remain underexplored.

Model Merging: Combining models to integrate specialized capabilities has gained traction through methods like Task Arithmetic (Ilharco et al., 2022), TIES (Yadav et al., 2024), and Model Breadcrumbs (Davari & Belilovsky, 2023). We adopt Task Arithmetic to extract instruction residuals, though analyzing how different merging strategies affect capability transfer is left for future work.

6. Conclusion

In conclusion, this study delves into the effects of continuous pre-training on base and instruction-tuned large language models (LLMs) and their instruction capabilities. The findings suggest that while continuous pre-training of instruction models may lead to catastrophic forgetting of instruction capabilities, a more efficient approach is to continuously pre-train the base model with new data, followed by instruction tuning. This method preserves both domain knowledge and instruction capabilities. Interestingly, the study also reveals that instruction capabilities are transferable across models from the same ancestor, eliminating the need for additional instruction tuning.

References

Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Caccia, M., Rodriguez, P., Ostapenko, O., Normandin, F., Lin, M., Caccia, L., Laradji, I., Rish, I., Lacoste, A., Vazquez, D., et al. Online fast adaptation and knowledge accumulation: a new approach to continual learning. *arXiv preprint arXiv:2003.05856*, 2020.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dallabetta, M., Dobberstein, C., Breiding, A., and Akbik, A. Fundus: A simple-to-use news scraper optimized for high quality extractions. In Cao, Y., Feng, Y., and Xiong, D. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pp. 305–314, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-demos.29.

Davari, M. and Belilovsky, E. Model breadcrumbs: Scaling multi-task model merging with sparse masks. *arXiv preprint arXiv:2312.06795*, 2023.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Gao, L., Tow, J., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., McDonell, K., Muennighoff, N., et al. A framework for few-shot language model evaluation. *Version v0. 0.1. Sept*, 10:8–9, 2021.

Gogoulou, E., Lesort, T., Boman, M., and Nivre, J. A study of continual learning under language shift. *arXiv preprint arXiv:2311.01200*, 2023.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Hu, E. J., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Ibrahim, A., Thérien, B., Gupta, K., Richter, M. L., Anthony, Q., Lesort, T., Belilovsky, E., and Rish, I. Simple and scalable strategies to continually pre-train large language models. *arXiv preprint arXiv:2403.08763*, 2024.

Ilharco, G., Ribeiro, M. T., Wortsman, M., Gururangan, S., Schmidt, L., Hajishirzi, H., and Farhadi, A. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Jiang, M., Liu, K., Zhong, M., Schaeffer, R., Ouyang, S., Han, J., and Koyejo, S. Does data contamination make a difference? insights from intentionally contaminating pre-training data for language models. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2024. URL https://openreview.net/forum?id=wSpwj7xab9.

Kosec, M., Fu, S., and Krell, M. M. Packing: Towards 2x nlp bert acceleration. 2021.

Le Scao, T., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., et al. Bloom: A 176b-parameter open-access multilingual language model. 2023.

Lin, S., Hilton, J., and Evans, O. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.

Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., and Chen, M.-H. Dora: Weight-decomposed low-rank adaptation. *arXiv preprint arXiv:2402.09353*, 2024.

Ma, S., Huang, S., Huang, S., Wang, X., Li, Y., Zheng, H.-T., Xie, P., Huang, F., and Jiang, Y. Ecomgpt-ct: Continual pre-training of e-commerce large language models with semi-structured data. *arXiv preprint arXiv:2312.15696*, 2023.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Roziere, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Sun, Z., Shen, Y., Zhou, Q., Zhang, H., Chen, Z., Cox, D., Yang, Y., and Gan, C. Principle-driven self-alignment of language models from scratch with minimal human supervision. *Advances in Neural Information Processing Systems*, 36, 2024.

Team, G., Mesnard, T., Hardin, C., Dadashi, R., Bhupatiraju, S., Pathak, S., Sifre, L., Rivière, M., Kale, M. S., Love, J., et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL `https://qwenlm.github.io/blog/qwen2.5/`.

Tokpanov, Y., Millidge, B., Glorioso, P., Pilault, J., Ibrahim, A., Whittington, J., and Anthony, Q. Zyda: A 1.3 t dataset for open language modeling. *arXiv preprint arXiv:2406.01981*, 2024.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Wang, Y., Mishra, S., Alipoormolabashi, P., Kordi, Y., Mirzaei, A., Naik, A., Ashok, A., Dhanasekaran, A. S., Arunkumar, A., Stap, D., et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5085–5109, 2022.

Wang, Y., Ma, X., Zhang, G., Ni, Y., Chandra, A., Guo, S., Ren, W., Arulraj, A., He, X., Jiang, Z., et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*, 2024.

Xie, Y., Aggarwal, K., and Ahmad, A. Efficient continual pre-training for building domain specific large language models. *arXiv preprint arXiv:2311.08545*, 2023.

Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., and Jiang, D. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.

Yadav, P., Sun, Q., Ding, H., Li, X., Zhang, D., Tan, M., Ma, X., Bhatia, P., Nallapati, R., Ramanathan, M. K., et al. Exploring continual learning for code generation models. *arXiv preprint arXiv:2307.02435*, 2023.

Yadav, P., Tam, D., Choshen, L., Raffel, C. A., and Bansal, M. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yang, A., Yang, B., Hui, B., Zheng, B., Yu, B., Zhou, C., Li, C., Li, C., Liu, D., Huang, F., et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024a.

Yang, X., Gao, J., Xue, W., and Alexandersson, E. Pllama: An open-source large language model for plant science. *arXiv preprint arXiv:2401.01600*, 2024b.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Zhou, J., Lu, T., Mishra, S., Brahma, S., Basu, S., Luan, Y., Zhou, D., and Hou, L. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

A. Instruction Portability Quality Comparison

B. Detailed Experiment Settings

B.1. Datasets

B.1.1. Pre-traininig Dataset

We need this pre-training dataset to test the impact on instruction capabilities of the continuously pre-trained model.

Table 4: Instruction portability quality comparison with LLaMa 3 and 3.1 8B LLMs. Where the **BOLD** represents the best quality and the underline shows the second-best score for that column block.

LLM (Params) →		**LLaMa 3 (8B)**			**LLaMa 3.1 (8B)**			**Qwen 2 (1.5B)**			**Qwen 2.5 (1.5B)**		
Benchmark	Metric	L3b	L3i	+3.1Lr	L3.1b	L3.1i	+3Lr	Q2b	Q2i	+2.5Qr	Q2.5b	Q2.5i	+2Qr
IFEval	ILL_acc	19.06	53.36	**57.19**	15.59	**54.80**	48.32	25.84	**29.02**	27.22	28.15	**39.09**	26.50
	ILS_acc	17.87	47.84	**51.92**	14.63	**49.88**	43.76	23.62	**26.74**	23.86	25.72	**35.21**	25.06
	PLL_acc	09.98	41.77	**43.81**	07.76	**41.59**	34.38	16.08	**17.93**	17.19	20.01	**25.67**	20.33
	PLS_acc	09.06	35.30	**37.52**	07.02	**35.49**	29.39	14.05	**15.71**	14.23	17.65	**22.04**	18.85
MMLU	acc	62.14	63.83	**66.02**	63.40	**68.03**	64.89	55.10	**55.80**	55.16	59.75	**59.75**	59.75
MMLU-Pro	EM	34.51	**39.70**	38.53	35.32	**40.97**	40.40	21.21	**21.70**	21.10	27.13	**29.89**	27.00
GSM8K	EM	49.58	75.06	**75.13**	50.11	**76.19**	73.84	54.51	**58.30**	57.01	57.39	56.86	**58.76**
	strict-EM	49.20	74.98	**74.98**	49.81	**75.36**	73.84	54.44	**57.39**	56.79	57.16	53.37	**58.30**
Sub-Average		31.43	53.98	**55.64**	30.46	**55.29**	51.10	33.11	**35.32**	34.07	36.62	**40.24**	36.82
Winogrande	acc	**73.16**	71.74	72.77	**73.64**	73.40	73.01	**66.38**	65.19	65.82	63.22	**65.19**	63.22
Hellaswag	acc	**60.12**	57.70	59.00	**60.02**	59.10	58.25	48.61	**49.30**	48.58	50.16	**50.94**	50.08
	acc_n	**79.22**	75.76	78.88	78.90	**79.19**	76.68	65.43	**66.07**	65.42	67.81	**68.34**	67.76
ARC_easy	acc	80.39	81.52	**81.57**	81.40	81.94	**82.91**	66.25	**69.99**	66.37	75.42	**76.56**	75.67
	acc_n	77.78	**79.63**	78.83	81.14	79.76	**81.86**	60.86	**66.54**	60.14	71.84	**76.26**	72.26
Piqa	acc	**79.54**	78.56	79.00	**80.09**	80.03	79.16	75.41	**76.17**	75.95	75.68	**76.39**	75.84
	acc_n	**80.74**	78.62	80.14	81.07	**81.12**	79.22	75.41	**75.90**	75.79	76.06	**76.12**	75.79
Sub-Average		**75.85**	74.79	75.74	**76.61**	76.36	75.87	65.48	**67.02**	65.48	68.60	**69.97**	68.66
T_mc2	acc	43.94	51.67	**52.73**	45.17	**53.92**	52.20	45.95	43.36	**45.95**	**46.64**	46.65	46.45
Average		51.64	62.94	**64.25**	51.57	**64.42**	62.01	48.07	**49.69**	48.54	51.24	**53.65**	51.35

We want the new pre-training data such that none of the base models and the corresponding instruct model have seen that data previously. Since the data contamination is a serious concern as noted in Jiang et al. (2024), the existing pre-training datasets may not be the right choice to continuously pre-train the model. Therefore, we manually scraped around 2M articles using a static news crawler FUNDUS[1] (Dallabetta et al., 2024).

We choose the news articles that are new to LLaMa 3.1 models that is we choose the articles published in the date range from December 2023 to September 2024 from all existing publishers in FUNDUS. The average length of the articles is 650 LLaMa tokens with 6981 max tokens and 156 min tokens. These articles are then packed with a sequence length of 4096 (LLaMa maximum sequence length is 8K but we choose 4K to efficiently utilize the existing GPU vRAM), and similar to Kosec et al. (2021) we use attention masks for each article to avoid cross article contamination.

B.1.2. Evaluation Dataset

In this section, we describe the test dataset used to evaluate our hypothesis. To perform a comprehensive evaluation

[1] https://github.com/flairNLP/fundus

Table 5: Evaluation dataset categorization.

Category	Sub Category	Benchmark
Instruction following	Language understanding	IFEval
		MMLU MMLU-Pro
	Math and logic	GSM8K
Reasoning and problem solving	Commonsense	Winogrande Hellaswag
	Factual knowledge	ARC_easy
	Physical reasoning	Piqa
Truthfulness		Truthfulqa_mc2

and to maintain reproducibility we use the evaluation harness framework from EleutherAI (Gao et al., 2021). We particularly target to evaluate the following capabilities:

Instruction following

IFEval focuses on natural language instruction following capabilities of LLMs (Zhou et al., 2023). It contains 25 types of verifiable instructions such as *write in more than 400 words*, *mention the keyword of AI at least 3 times* with 500 prompts. This evaluation is performed on 4 metrics:

(1) Prompt-level strict-accuracy (PLS-acc): The percentage of prompts that all verifiable instructions in each prompt are followed, (2) Inst-level strict-accuracy (ILS-acc): The percentage of verifiable instructions that are followed, (3) Prompt-level loose-accuracy (PLL-acc): Prompt-level accuracy computed with the loose criterion, and (4) Inst-level loose-accuracy (ILL-acc): Instruction-level accuracy computed with a loose criterion.

MMLU mainly focuses on extensive world knowledge across 57 subjects which includes all major domains like math, computer science, medicine, philosophy, and law (Hendrycks et al., 2020). This dataset contains a total of 15908 development and test questions with 4 possible answers each.

MMLU-Pro is introduced to further increase the complexity of the MMLU benchmark since the existing LLMs are excelled at MMLU (Wang et al., 2024) by eliminating some trivial and noisy questions from MMLU and by introducing reasoning-focused questions to MMLU which has mostly knowledge-driven questions.

GSM8K dataset consists of 8.5K high-quality linguistically diverse grade school math problems (Cobbe et al., 2021). These problems take between 2 and 8 steps to solve, and solutions primarily involve performing a sequence of elementary calculations using basic arithmetic operations ($+, -, \times, \div$) to reach the final answer.

Reasoning and problem-solving

Winogrande is a large scale 44k commonsense reasoning dataset. It mainly tests a model's ability to resolve ambiguous pronouns based on contextual understanding (Sakaguchi et al., 2021).

Hellaswag is designed to benchmark commonsense reasoning in AI models (Zellers et al., 2019). It contains 10,000 multiple-choice questions for validation and testing. The dataset focuses on predicting the most plausible continuation of a given scenario.

ARC_easy dataset consists of a collection of 7787 natural science questions (Clark et al., 2018). The dataset contains only natural, grade-school science questions. ARC questions appeal to both different styles of knowledge and different styles of reasoning.

Piqa evaluates the model on the physical commonsense questions without experiencing the physical world (Bisk et al., 2020). Each instruction has a goal to reach in the physical world, given the description of the environment if required, and 2 options (solutions) to reach the goal.

Truthfulness measure the truthfulness of a language model in answering questions (Lin et al., 2021). This dataset consists of 817 questions across 38 categories and captures human misconceptions, false beliefs, conspiracies, and awareness between real-world knowledge and fictional knowledge across 38 domains including health, law, finance, and politics. Because of space constraints, we abbreviate this dataset as *T_mc2*.

We choose these datasets as these are commonly evaluated for most of the newly released LLMs (Touvron et al., 2023; Yang et al., 2024a). Table 5 summarizes all the evaluation datasets used in this work for each category. We used the latest versions of these datasets available on EleutherAI[2] as of writing this work. Only MMLU, MMLU-Pro, and GSM8K are evaluated with 5-shot, rest of the datasets are evaluated on zero-shot.

B.2. Language Model Architectures

We used two distinct families of language models LLaMa (Dubey et al., 2024) and Qwen (Yang et al., 2024a). Specifically, we target the LLaMa 3, 3.1 family of models, and the Qwen 2, 2.5 family of models. Both LLaMa 3 and 3.1 are available in 8B, 70B parameters size with the exception that the 3.1 family also has a 405B parameters model. For all our LLaMa experiments we focused only on the 8B models because of the resource constraints. Similarly, both Qwen 2 and 2.5 come in 0.5B, 1.5B, and 7B parameter models with the exception that the 2.5 family also has 3B, 14B, 32B, and 72B parameter models. For all Qwen experiments, we choose 0.5B, 1.5B, and 7B models. Further, by design, we are required to choose a family of models for which both the base and the instruction-tuned variants of the same size exist.

C. Overview of Experimental Hardware

C.1. GPU Specifications

- GPU: NVIDIA A100 40GB SXM
- GPU Memory: 40GB
- FP16/BF16 Tensor Core: 312 TeraFLOPs
- TF32: 156 TeraFLOPs

C.2. FLOPs Requirements

Instruction Fine-tuning

- Number of Parameters, N: 8B.

[2] `https://github.com/EleutherAI/lm-evaluation-harness`

- Number of tokens, $tokens$

$$\begin{aligned} tokens &: 25M samples \\ &\approx 25M \times 8192 \\ &= 204,800\text{M tokens} \end{aligned}$$

- Number of Epochs, E: Fine-tuning generally requires fewer epochs; often 3 to 10 epochs are sufficient.

Continued Pre-training The below calculations assume continuous pre-training with 100M Tokens.

- Number of Parameters, N: 8B.
- Number of tokens, $tokens$: 100M tokens
- Sequence Length, S: 4096
- Number of Epochs, E: 5

Estimate FLOPs per Tokens

The FLOPs per training step depend on the number of operations performed per token per layer. Assuming each parameter needs about 6 floating-point operations (forward and backward):

$$\text{FLOPs/token/parameter} \approx 6$$

Given the structure of transformers with multiple layers and self-attention, let's simplify and assume each token requires 6 operations per parameter across all layers:

$$\text{FLOPs/token} = 6 \times N$$

C.3. A Comparison

Here, we perform the comparison between LLaMa8B Instruction Tuning and 100M continuous pre-training (CP) in terms of the number of FLOPs.

$$\begin{aligned} ratio &= \frac{\text{Instruct}(6 \times 8 \times 10^9 \times tokens \times E)}{\text{CP}(6 \times 8 \times 10^9 \times tokens \times E)} \\ &= \frac{6 \times 10^9 \times 204800 Million \times 5}{6 \times 10^9 \times 100 Million \times 5} \\ &\approx 2048 \end{aligned}$$

This calculation provides a rough estimate of the FLOPs required to continue pre-training the model on 100M tokens across 5 epochs and instruction fine-tuning FLOPs, estimates numbers from Llama3 Paper.

D. MMLU Performance vs Compute Cost

we demonstrate how our model maintained a good enough performance on the MMLU benchmark while optimizing for low computational costs. The results reflect efficient usage of available compute resources, particularly by leveraging hardware such as the NVIDIA A100.

D.1. Comparison of MMLU Scores and Compute Costs

Our approach achieved high accuracy in various tasks under the MMLU benchmark, matching the performance of more compute-intensive models. Despite this, we successfully reduced the total compute cost by optimizing training and fine-tuning processes, as shown in Figure 2.

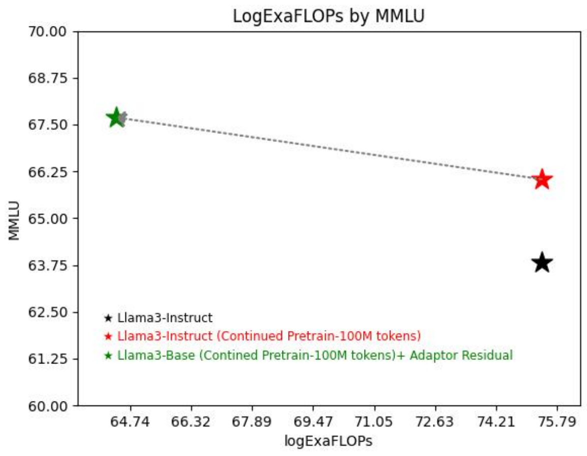

Figure 2: Comparison of MMLU performance and compute costs across different models. Our model (marked in green) balances compute efficiency while maintaining competitive performance.

E. A Note on Notation

- `L3b`: is the LLaMa 3 base model $\left(\theta_{\text{L3b}}^{d_1}\right)$.
- `L3i`: is the LLaMa 3 instruct model $\left(\theta_{\text{L3i}}^{d_1 v_1}\right)$.
- `3Lr`: is the instruction residuals from LLaMa 3 $\left(\theta_{\text{L3i}}^{d_1 v_1} - \theta_{\text{L3b}}^{d_1}\right)$.
- `org`: means the corresponding model with no domain adaptation
- `+100M for L3b`: means `L3b` is continuously fine-tuned for 100M new tokens.
- `+100M for L3i`: means `L3i` is continuously fine-tuned for 100M new tokens.

- `100M for L3b+3Lr`: means `L3b` is continuously finetuned for 100M new tokens and the instruction capability is added via insturction residuals `3Lr`.

Table 6: Key Notations

Symbol	Meaning
θ_i	Instruction-tuned LLM Params
θ_b	Pretrained base LLM Params
θ_i^x, θ_b^y	Instruction/base model trained on datasets x/y
Θ_r^z	Instruction residual: $\theta_i^x - \theta_b^y$
$\theta_b^x + \Theta_r^z$	Reconstructed instruction model
	Specific notation meaning
$\theta_b^{d_1}$	Base LLM pretrained on d_1 dataset
$\theta_b^{d_1 d_2}$	Base LLM pretrained on d_1 dataset and then continuous pretrained on d_2 dataset
$\theta_i^{d_1 v_1}$	Base LLM pretrained on d_1 dataset and then instruction fine-tuned on v_1 dataset
$\theta_i^{d_1 d_2 v_1}$	Base LLM pretrained on d_1 dataset, continuous pretrained on d_2 dataset and instruction fine tuned on v_1
$\theta_i^{d_1 v_1 d_2}$	Base LLM pretrained on d_1 dataset, instruction fine tuned on v_1 and further continuous pre-trained on d_2 dataset.