Instruction Tuning With Loss Over Instructions

Anonymous Authors[1]

Abstract

Instruction tuning plays a crucial role in shaping the outputs of language models (LMs) to desired styles. In this work, we propose a simple yet effective method, INSTRUCTION MODELLING (IM), which trains LMs by applying a loss function to the instruction and prompt part rather than solely to the output part. Through experiments across 21 diverse benchmarks, we show that, in many scenarios, IM can effectively improve the LM performance on both NLP tasks (*e.g.,* MMLU, TruthfulQA, and HumanEval) and open-ended generation benchmarks (*e.g.,* MT-Bench and AlpacaEval). Remarkably, in the most advantageous case, IM boosts model performance on AlpacaEval 1.0 by over 100%. We identify two key factors influencing the effectiveness of IM: (1) The ratio between instruction length and output length in the training data; and (2) The number of training examples. We observe that IM is especially beneficial when trained on datasets with lengthy instructions paired with brief outputs, or under the Superficial Alignment Hypothesis (SAH) where a small amount of training examples are used for instruction tuning. Further analysis substantiates our hypothesis that the improvement can be attributed to reduced overfitting to instruction tuning datasets. Our work provides practical guidance for instruction tuning LMs, especially in low-resource scenarios. Our code is available at `https://anonymous.4open.science/r/InstructionModelling-2632`.

1. Introduction

them to learn general-purpose representations transferable to various language understanding or generation tasks. However, it does align LMs to act in accordance with the user's intentions [28]. To enable this transfer, various methods for aligning language models have thus been proposed, one of which is instruction tuning (IT) [38, 2, 9]. Recent study [62] proposes Superficial Alignment Hypothesis (SAH): A model's knowledge and capabilities are learnt almost entirely during pretraining, only minimal instruction tuning data is required to enable high-quality outputs in the desired output style. Existing works [1, 37, 38, 43, 53, 55, 30] mainly perform instruction tuning by focusing the loss computation solely on the output segments.

In this work, we demonstrate that incorporating an additional loss component for instructions or prompts, which we refer to as INSTRUCTION MODELLING (IM) (see §2), could substantially improve the performance of instruction tuning on both various NLP tasks (*e.g.,* MMLU, TruthfulQA, and HumanEval) and open-ended generation benchmarks (*e.g.,* MT-Bench and AlpacaEval), as shown in Figure 1. Remarkably, in the most favourable case, our proposed method IM boosts performance on AlpacaEval 1.0 by over 100%. As illustrated in Figure 2, Our study further identifies two key factors influencing the effectiveness of IM: (1) **The ratio between instruction length and output length** (see Figure 2 Left). Our analysis shows that our approach IM is especially beneficial for datasets characterised by lengthy instructions or prompts paired with comparably brief outputs, such as `Code Alpaca` [8] and `Less MMLU Chat` [54]. (2) **The number of training examples** (see Figure 2 Right). We demonstrate that our approach IM performs better under the low-resource setting or SAH, where fewer training examples are available (§3.2).

We hypothesise that the improvement stems from reducing instruction tuning's tendency to overfit, particularly under limited training resource conditions. Recent works [22, 25, 38, 57, 58] suggest that LMs can quickly memorise training examples even after seeing them just once. Training on a small amount of instruction tuning data for a few epochs can potentially lead to rapid overfitting. To substantiate our hypothesis, our analysis shows that (1) IM exhibits higher training losses but lower test losses on new instruction tuning data; (2) The outputs generated by IM have a lower similarity to the training examples compared to those from IT, as indicated by BLEU scores; and (3) IM has less instruction tuning tax on NLP tasks across

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the ICML 2024 Workshop on Foundation Models in the Wild. Do not distribute.

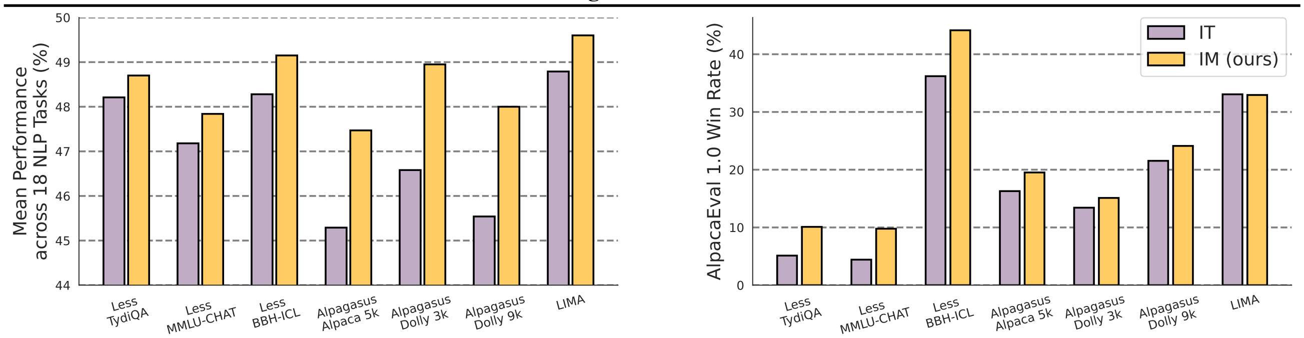

Figure 1. Performance differences between INSTRUCTION TUNING (IT) and our proposed method INSTRUCTION MODELLING (IM) trained on 7 instruction tuning datasets. (**Left**) The mean performance across 18 traditional NLP tasks. (**Right**) The win rate on the AlpacaEval 1.0 benchmark.

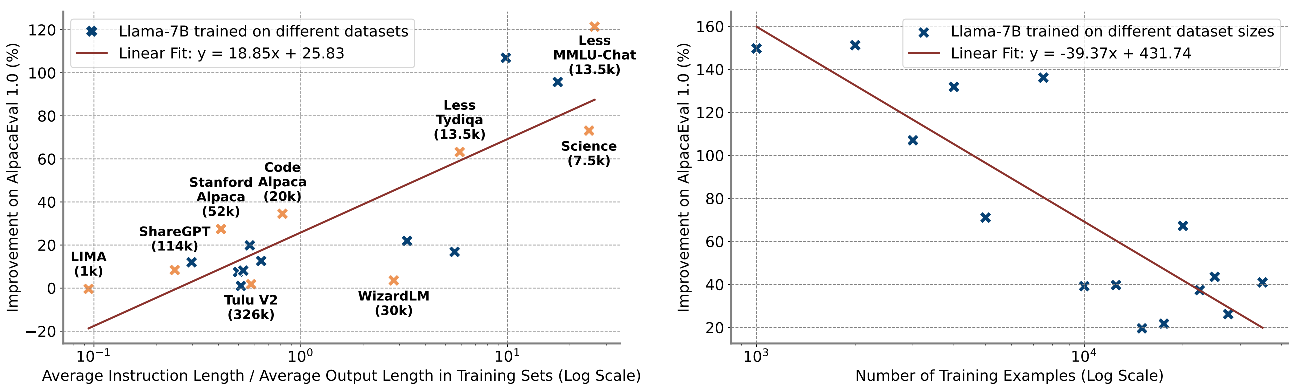

Figure 2. (**Left**) Performance improvement, achieved by our approach INSTRUCTION MODELLING (IM) compared to INSTRUCTION TUNING (IT) on the AlpacaEval 1.0, against the ratio between average instruction length and average output length in instruction tuning datasets (training size noted in parentheses). (**Right**) Performance improvement achieved by our approach IM over IT on the AlpacaEval 1.0 against the number of training examples in instruction tuning datasets.

training epochs (§B). Additionally, our study reveals that this overfitting cannot be effectively addressed by applying Kullback-Leibler (KL) divergence for regularisation, as it compromises the model's ability to follow instructions. Our further analysis reveals that the advantages of IM persist across different LMs and model sizes, and that IM could be effectively combined with the previous approach (*i.e.,* NEFTUNE [25]). Meanwhile, we investigate the relationship between output length and win rate for our approach (§C).

In summary, the main contributions of this paper are:

- We propose INSTRUCTION MODELLING (IM), aiming to enhance both the instruction-following and general performance on NLP tasks of LMs. Through extensive experiments across 21 benchmarks, we demonstrate that, in many scenarios, IM substantially improves performance of LMs trained on various instruction tuning datasets, particularly notable in the AlpacaEval 1.0 benchmark where it boosts scores by over 100%.
- Our study identifies key factors influencing the effectiveness of IM, including the ratio between instruction length and output length and the number of training examples, providing practical guidance for instruction tuning LMs, especially under the low-resource scenarios.
- We provide underlying mechanisms that make IM effective, specifically how it mitigates overfitting, thereby enhancing the LMs' performance across various tasks.

2. Our Approach

Let x be the full sequence including any template tokens T, which are ignored in the loss calculation. The loss function $\mathcal{L}$ calculates the negative log-likelihood for both instruction and completion tokens, excluding any prompt template tokens:

$$\mathcal{L} = -\sum_{t=1}^{m+n} \log P(x_t | x_1, x_2, ..., x_{t-1}) \cdot \mathbf{1}(x_t \notin T), \quad (1)$$

where $\mathbf{1}(x_t \notin T)$ is an indicator function that is 1 if x_t is not a template token and 0 otherwise. This ensures that the loss is computed only over the meaningful tokens, not over the static template tokens.

3. Experiments and Results

In this section, we evaluate the effectiveness of our proposed method INSTRUCTION MODELLING (IM) by comparing it with INSTRUCTION TUNING (IT) and other baselines on various datasets.

3.1. Experimental Setup

Instruction Tuning Datasets. We assess our method, IM, across various instruction tuning datasets, detailed as follows: (1) `Stanford Alpaca` [49] (52,002 examples); (2) `Dolly` [13] (15,011 examples); (3) `Sharegpt` [8] (50,000 examples); (4) `Code Alpaca` [5] (20,022 examples); (5) `Science Literature` [24] (7,544 examples); (6) `WizardLM` [55] (30,000 examples); (7) `Tulu V2` [24] (326,181 examples). Additionally, we incorporate instruction tuning datasets under the low-resource setting or SAH: (8) `LIMA` [62] (1,030 examples); (9) `Less`[1] [54], where high-quality instruction tuning data are selected from `Flan V2` and `Dolly`. Here, we use the `Less MMLU Chat` (13,533 examples), `Less BBH ICL` (13,533 examples), and `Less Tydiqa` (13,533 examples); (10) `Alpagasus`[2] [6], which offers three subsets: `Alpagasus Dolly 3k` (2,996 examples), `Alpagasus Dolly 9k` (9,229 examples) selected from `Dolly`, and `Alpagasus Alpaca 5k` (5,305 examples) selected from `Stanford Alpaca`. See dataset details and statistical analysis in Appendix §D.

Evaluation Benchmarks. Our study conducts a comprehensive analysis of 21 NLP datasets, focusing on a suite of canonical NLP benchmarks and their capacity for open-ended language generation. For canonical NLP benchmarks, the evaluation is organised into six categories (18 tasks in total): (1) *Language Understanding and Knowledge* includes MMLU [19], PIQA [3], OpenbookQA [35], HellaSwag [59], and LAMBADA [39]; (2) *Multilinguality* contains LAMBADA Multilingual [39], WMT 2014 [4], and WMT 2016 [44]; (3) *Commonsense Reasoning* features Winograd schema challenge (WSC) [29], WinoGrande [42], AI2 Reasoning Challenge (ARC) [11], and CoQA [41]; (4) *Math and Coding Reasoning* includes GSM8K [12], and HumanEval [7]; (5) *Safety and Helpfulness* comprises TruthfulQA [32], ToxiGen [16], and Hendrycks Ethics [18]. (6) *Big Bench Hard (BBH)* dataset [48] is included to assess models. Our models are also tested for their open-ended text generation capabilities using model-based evaluations, specifically through MT-Bench [61], AlpacaEval 1.0 and AlpacaEval 2.0 [31], where the AlpacaEval 1.0 compares the model outputs against the `text_davinci_003` evaluated by GPT-4 and the AlpacaEval 2.0 compares the model outputs against `GPT-4` outputs evaluated by GPT-4 Turbo. See evaluation details in Appendix §E.

[1] `https://github.com/princeton-nlp/LESS`
[2] `https://github.com/gpt4life/alpagasus`

All Comparison Approaches. In our study, we mainly experiment using the LLAMA-2-7B-BASE and LLAMA-2-13B-BASE [50], and the OPT-6.7B [60] models. We report model performance trained on LLAMA-2-7B-BASE if not specified. We compare with NEFTUNE [25] as the baseline, which adds noise to the embedding during the instruction tuning to increase the robustness of instruction-tuned models. In this paper, we use several dataset selection papers. See hyperparameter and implementation details in Appendix §F.

3.2. Main Results

In this section, we first evaluate the model performance of our approach and baselines across various tasks. Then we investigate the key factors that contribute to the effectiveness of our approach. Below we will discuss our findings.

#1: Our approach IM can improve the performance of Instruction Tuning on various NLP tasks and open-ended generation benchmarks. Figure 1 provides a summary of the model's performance across both traditional NLP tasks and the AlpacaEval 1.0 benchmark. Table 1 offers a detailed breakdown of experimental results for traditional NLP tasks across six categories, as well as performance on additional benchmarks for open-ended generation (*i.e.,* MT-Bench and AlpacaEval). The experimental results show that our approach IM can improve the performance of instruction tuning on various NLP tasks and open-ended generation benchmarks. Specifically, on the `Alpagasus Dolly 3k` dataset, IM improves the overall mean score of NLP tasks to 48.95, an increase of 2.37 points from the baseline. Similarly, on the `Alpagasus Dolly 9k` dataset, we observe an improvement of 2.46 points in the mean NLP score.

#2: Our approach IM is especially beneficial for datasets characterised by lengthy instructions or prompts paired with comparably brief outputs. To better understand the impact factors on the effectiveness of IM, we extend our experiments to more instruction-tuning datasets, such as `Science Literature`, `Code Alpaca` and `Tulu V2`. Interestingly, as shown in Figure 2 Left, we find that IM is particularly effective in scenarios where datasets characterised by lengthy instructions and shorter outputs, such as `Less MMLU Chat` and `Less BBH ICL`. For example, in datasets like `Less MMLU Chat` and `Less Tydiqa`, IM shows remarkable efficacy. In contrast, the

Table 1. Performance comparisons using 7 instruction tuning datasets with the LLAMA-2-7B on 6 categories of 23 traditional NLP tasks and 3 open-ended benchmarks with LLM as judgements. "IT" refers to INSTRUCTION TUNING. "IM" refers to INSTRUCTION MODELLING. Green and red arrows indicate performance changes against the baseline (IT).

	NLP Benchmarks							LLM-based Evaluation		
Method	Understanding & Knowledge	Multi-linguality	Commonsense Reasoning	Math&Code Reasoning	BBH	Safety & Helpfulness	Mean	MT-Bench	AlpacaEval 1.0	AlpacaEval 2.0
LLAMA-2-BASE	63.91	61.99	75.86	13.32	38.80	42.03	49.32	1.16	0.01	0.01
LLAMA-2-CHAT	63.42	55.15	70.28	15.33	38.92	51.79	49.15	6.63	79.04	6.48
`Alpagasus Alpaca 5k` (5,305 training examples)										
IT	64.98	57.24	66.06	8.93	26.80	47.74	45.29	3.62	16.29	2.46
NEFTUNE	65.18	56.88	66.45	10.24	29.53	45.46	45.62 $\uparrow$0.33	3.50 $\downarrow$0.12	21.37 $\uparrow$5.08	2.37 $\downarrow$0.09
IM (ours)	64.01	56.63	72.47	11.58	35.52	44.62	47.47 $\uparrow$2.18	3.48 $\downarrow$0.14	19.52 $\uparrow$3.23	3.29 $\uparrow$0.83
`Alpagasus Dolly 3k` (2,996 training examples)										
IT	65.81	57.46	67.55	11.96	33.02	43.70	46.58	4.23	13.42	2.00
NEFTUNE	65.90	57.79	67.28	11.64	35.43	44.36	47.07 $\uparrow$0.49	4.42 $\uparrow$0.19	14.04 $\uparrow$0.62	2.03 $\uparrow$0.03
IM (ours)	65.66	57.47	73.24	14.57	37.48	45.29	48.95 $\uparrow$2.37	4.06 $\downarrow$0.17	15.11 $\uparrow$1.69	2.44 $\uparrow$0.44
`Alpagasus Dolly 9k` (9,229 training examples)										
IT	64.10	56.62	69.70	7.96	32.19	42.65	45.54	4.33	21.54	2.28
NEFTUNE	64.20	56.69	69.51	8.99	33.91	42.62	45.99 $\uparrow$0.45	4.21 $\downarrow$0.12	31.61 $\uparrow$10.07	2.84 $\uparrow$0.56
IM (ours)	64.67	55.32	74.87	12.50	36.69	43.96	48.00 $\uparrow$2.46	4.55 $\uparrow$0.22	30.77 $\uparrow$9.23	2.67 $\uparrow$0.39
`Less Tydiqa` (13,533 training examples)										
IT	64.01	56.81	64.77	12.06	36.54	55.09	48.21	4.08	5.12	1.88
NEFTUNE	64.03	55.09	64.02	13.84	36.65	51.21	47.47 $\downarrow$0.74	4.19 $\uparrow$0.11	8.35 $\uparrow$3.23	2.58 $\uparrow$0.70
IM (ours)	64.28	56.10	65.70	17.15	34.86	54.09	48.70 $\uparrow$0.49	4.36 $\uparrow$0.28	10.10 $\uparrow$4.98	2.88 $\uparrow$1.00
`Less MMLU Chat` (13,533 training examples)										
IT	64.74	57.42	62.94	9.53	33.13	55.35	47.18	3.86	4.42	1.20
NEFTUNE	65.21	57.43	63.14	9.45	35.89	55.32	47.74 $\uparrow$0.56	4.06 $\uparrow$0.20	6.22 $\uparrow$1.80	1.06 $\downarrow$0.14
IM (ours)	63.95	56.34	64.76	12.52	36.94	52.55	47.84 $\uparrow$0.66	4.54 $\uparrow$0.68	9.78 $\uparrow$5.36	1.93 $\uparrow$0.73
`Less BBH ICL` (13,533 training examples)										
IT	63.83	62.04	75.92	6.90	38.93	42.07	48.28	4.78	36.20	2.36
NEFTUNE	63.88	58.83	67.97	13.54	38.63	51.33	49.03 $\uparrow$0.75	5.05 $\uparrow$0.27	39.81 $\uparrow$3.61	2.87 $\uparrow$0.51
IM (ours)	64.14	56.72	71.12	13.56	39.03	50.34	49.15 $\uparrow$0.87	5.03 $\uparrow$0.25	44.15 $\uparrow$7.95	3.56 $\uparrow$1.20
`LIMA` (1,030 training examples)										
IT	63.92	58.29	71.96	16.01	39.27	43.29	48.79	4.77	33.06	2.58
10 epoch NEFTUNE	63.66	57.67	73.03	15.95	38.77	43.14	48.70 $\downarrow$0.09	4.79 $\uparrow$0.02	30.51 $\downarrow$2.55	2.43 $\downarrow$0.15
IM (ours)	64.49	58.21	75.55	17.06	38.84	43.45	49.60 $\uparrow$0.81	4.83 $\uparrow$0.06	32.94 $\downarrow$0.12	2.47 $\downarrow$0.11

`Tulu V2` dataset, with an instruction to output length ratio of about 0.5, benefits less compared to the `Science Literature` dataset, which has a much higher ratio of 24.7. We hypothesise that this trend can be attributed to the tendency of language models trained on datasets with shorter outputs to overfit. In cases where the instructions are longer, IM acts as an effective form of regularisation, mitigating this issue. For further details on the experimental setup, refer to the Appendix in §F.

#3: Our approach IM performs better with fewer training examples. We find that another important factor in the effectiveness of IM is the quantity of training examples. Specifically, we design additional experiments by sampling different numbers of examples from the `Tulu V2` datasets, which contain about 320k training examples and achieve a modest improvement compared to other datasets in Figure 2 Left. We ensure that our samples maintain an instruction-to-output length ratio of around 10. As shown in Figure 2 Right, IM demonstrates substantial performance improvements on the AlpacaEval 1.0 as the number of training examples decreases. This suggests that IM could be particularly valuable for developing robust models in resource-constrained scenarios or under the SAH. For details on the experimental setup, please refer to the Appendix in §F.

4. Conclusion

In conclusion, our study proposes INSTRUCTION MODELLING, which trains LMs with loss over instructions rather than outputs only. Our experimental evaluations demonstrate that our approach largely improves the performance of LMs on both NLP tasks and open-ended generation benchmarks in some scenarios, especially under the Superficial Alignment Hypothesis and low-resource setting where minimal training data is used for instruction tuning. Our analysis has shed light on two key factors that influence the effectiveness of our approach, (1) the ratio between instruction and output lengths, and (2) the quantity of training data, providing practical insights for optimising instruction-based training methods. Our analysis reveals the mechanisms behind the effectiveness of IM, particularly its ability to reduce overfitting, showing that applying instruction losses in some scenarios can lead to more robust and adaptable LMs.

References

[1] Vamsi Aribandi, Yi Tay, Tal Schuster, Jinfeng Rao, Huaixiu Steven Zheng, Sanket Vaibhav Mehta, Honglei Zhuang, Vinh Q. Tran, Dara Bahri, Jianmo Ni, Jai Prakash Gupta, Kai Hui, Sebastian Ruder, and Donald Metzler. Ext5: Towards extreme multi-task scaling for transfer learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[2] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv preprint*, abs/2204.05862, 2022.

[3] Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press, 2020.

[4] Chris Callison-Burch, Philipp Koehn, Christof Monz, and Josh Schroeder. Findings of the 2009 Workshop on Statistical Machine Translation. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 1–28, Athens, Greece, 2009. Association for Computational Linguistics.

[5] Sahil Chaudhary. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

[6] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations*, 2024.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374, 2021.

[8] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023.

[9] 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. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[10] Kevin Clark, Minh-Thang Luong, Christopher D. Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, Brussels, Belgium, 2018. Association for Computational Linguistics.

[11] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv preprint*, abs/1803.05457, 2018.

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

[13] Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023.

[14] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023.

[15] Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. Textbooks are all you need. *ArXiv preprint*, abs/2306.11644, 2023.

[16] Thomas Hartvigsen, Saadia Gabriel, Hamid Palangi, Maarten Sap, Dipankar Ray, and Ece Kamar. ToxiGen: A large-scale machine-generated dataset for adversarial and implicit hate speech detection. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3309–3326, Dublin, Ireland, 2022. Association for Computational Linguistics.

[17] Guande He, Jianfei Chen, and Jun Zhu. Preserving pre-trained features helps calibrate fine-tuned language models. *ArXiv preprint*, abs/2305.19249, 2023.

[18] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning AI with shared human values. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[19] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[20] Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. Unnatural instructions: Tuning language models with (almost) no human labor. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428, Toronto, Canada, 2023. Association for Computational Linguistics.

[21] Tom Hosking, Phil Blunsom, and Max Bartolo. Human feedback is not gold standard. In *The Twelfth International Conference on Learning Representations*, 2024.

[22] Jeremy Howard and Jonathan Whitaker. Can llms learn from a single example?, 2023.

[23] Mathew Huerta-Enochian. Instruction fine-tuning: Does prompt loss matter?, 2024.

[24] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.

[25] Neel Jain, Ping yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. NEFTune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*, 2024.

[26] Aditi Jha, Sam Havens, Jeremey Dohmann, Alex Trott, and Jacob Portes. Limit: Less is more for instruction tuning across evaluation paradigms. *ArXiv preprint*, abs/2311.13133, 2023.

[27] Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online, 2020. Association for Computational Linguistics.

[28] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *ArXiv preprint*, abs/1811.07871, 2018.

[29] Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, Proceedings of the International Conference on Knowledge Representation and Reasoning, pages 552–561. Institute of Electrical and Electronics Engineers Inc., 2012. 13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012 ; Conference date: 10-06-2012 Through 14-06-2012.

[30] Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*, 2024.

[31] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023.

[32] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 3214–3252, Dublin, Ireland, 2022. Association for Computational Linguistics.

[33] Fangyu Liu, Qianchu Liu, Shruthi Bannur, Fernando Pérez-García, Naoto Usuyama, Sheng Zhang, Tristan Naumann, Aditya Nori, Hoifung Poon, Javier Alvarez-Valle, Ozan Oktay, and Stephanie L. Hyland. Compositional zero-shot domain transfer with text-to-text models. *Transactions of the Association for Computational Linguistics*, 11:1097–1113, 2023.

[34] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *ArXiv preprint*, abs/2312.15685, 2023.

[35] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium, 2018. Association for Computational Linguistics.

[36] Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. MetaICL: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809, Seattle, United States, 2022. Association for Computational Linguistics.

[37] Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487, Dublin, Ireland, 2022. Association for Computational Linguistics.

[38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[39] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany, 2016. Association for Computational Linguistics.

[40] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.

[41] Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019.

[42] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8732–8740. AAAI Press, 2020.

[43] Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal V. Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Févry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. Multitask prompted training enables zero-shot task generalization. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[44] Rico Sennrich, Barry Haddow, and Alexandra Birch. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany, 2016. Association for Computational Linguistics.

[45] Zhengxiang Shi, Yue Feng, and Aldo Lipani. Learning to execute actions or ask clarification questions. In

Findings of the Association for Computational Linguistics: NAACL 2022, pages 2060–2070, Seattle, United States, 2022. Association for Computational Linguistics.

[46] Zhengxiang Shi and Aldo Lipani. Don't stop pretraining? make prompt-based fine-tuning powerful learner. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[47] Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, et al. Aya dataset: An open-access collection for multilingual instruction tuning. *ArXiv preprint*, abs/2402.06619, 2024.

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

[49] 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, 2023.

[50] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *ArXiv preprint*, abs/2307.09288, 2023.

[51] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, 2023. Association for Computational Linguistics.

[52] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, Maitreya Patel, Mehrad Moradshahi, Mihir Parmar, Mirali Purohit, Neeraj Varshney, Phani Rohitha Kaza, Pulkit Verma, Ravsehaj Singh Puri, Rushang Karia, Savan Doshi, Shailaja Keyur Sampat, Siddhartha Mishra, Sujan Reddy A, Sumanta Patro, Tanay Dixit, and Xudong Shen. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics.

[53] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[54] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for instruction tuning, 2024.

[55] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024.

[56] Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. Rethinking the instruction quality: Lift is what you need, 2023.

[57] Fuzhao Xue, Yao Fu, Wangchunshu Zhou, Zangwei Zheng, and Yang You. To repeat or not to repeat: Insights from scaling LLM under token-crisis. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[58] Adam X Yang, Maxime Robeyns, Xi Wang, and Laurence Aitchison. Bayesian low-rank adaptation for large language models. In *ICLR*, 2024.

[59] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 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, 2019. Association for Computational Linguistics.

[60] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt:

Open pre-trained transformer language models. *ArXiv preprint*, abs/2205.01068, 2022.

[61] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[62] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Appendix Overview

The appendix is structured as follows:

Appendix §A presents the related works.

Appendix §B presents additional experiments showing that Instruction Modelling Mitigates Overfitting of Instruction Tuning.

Appendix §C presents additional analysis about instruction modelling.

Appendix §D provides a brief description (with statistical summaries) for instruction tuning datasets.

Appendix §E provides details of evaluation benchmarks and settings.

Appendix §F provides experimental setting, implementation details and hyperparameters for all comparison methods used in our experiments.

Appendix §G provides the supplementary experimental results to investigate the effect of our approach on training and testing losses.

Appendix §H provides the supplementary experimental results to investigate the relationship between the win rate on the AlpacaEval 1.0 and the number of epochs.

Appendix §I provides the mathematical formula for the Kullback-Leibler (KL) divergence used in our paper.

Appendix §J provides the supplementary experimental results to investigate the relationship between the output length and the number of epochs.

A. Related Work

Instruction Tuning. LMs can better align with user intents through fine-tuning on datasets consisting of instructions and human-written completions [2, 38]. Early studies mainly focus on NLP tasks, showing that fine-tuning with various NLP datasets trained with instruction output pairs improves cross-task generalisation [1, 27, 37, 38, 43, 46, 53]. Recent works explore the creation of instruction tuning datasets by LLMs themselves [51, 20, 55, 30] or through crowdsourcing approaches [8, 62]. Such instruction-tuning phrase [24, 45, 47, 21, 58] enables LLMs to generalise beyond instructions in the training set, largely enhancing their practical utility.

Data Selection for Instruction Tuning. Research on instruction tuning for LMs presents diverging perspectives on the optimal data scale for supervised fine-tuning. A prevailing view recommends fine-tuning on expansive datasets to enhance LM performance across various NLP tasks, thereby improving zero-shot and few-shot learning capabilities [1, 27, 38, 53, 37, 43, 52, 36]. For example, `Flan V2` comprises over a million question-answer pairs from diverse NLP sources [9], and `Natural Instructions` features 61 distinct tasks and 193k task instances [37]. Conversely, another research trajectory prioritises data quality over quantity [15, 56, 34, 26]. Superficial Alignment Hypothesis (SAH) [62] advocates for using smaller, high-quality datasets, arguing that LMs primarily acquire their capabilities during the pretraining phase and thus require only minimal data for effective instruction tuning. For instance, LIMA [62] employs a carefully curated set of 1k diverse prompts to generate stylistically consistent responses, aimed at creating a helpful AI assistant. AlpaGasus [6] and Less [54] employ methods to select high-quality data based on LLM-generated judgements and gradient signals, respectively. However, both views agree on the importance of (1) the quality of pre-trained base LMs and (2) the diversity and quality of the IT data.

Regularisation Through Language Modelling Objectives. Pretraining data and language modelling objectives have been used as a regularisation technique in fine-tuning LMs. In particular, [10, 33] fine-tunes LMs on labelled data, with unsupervised learning on unlabelled data for auxiliary tasks as regularisation. [38] mixes the alignment objective with the next token prediction objective using pretraining data to mitigate alignment tax in reinforcement learning from human feedback (RLHF). [17] adopts the masked language objective on the pretraining or downstream task corpus to preserve pre-trained features, and shows improvements in calibration and accuracy. [23] investigates the effect of incorporating instruction loss weighting on instruction tuning, suggesting that the instruction loss ratio is an important hyperparameter when fine-tuning short-completion data but is irrelevant when using long-completion data. In this work, we propose a broader guideline that does not introduce new hyperparameters but focuses on when and how to include loss over instruction effectively. We refer to our approach as INSTRUCTION MODELLING because it combines elements of both language modelling and instruction tuning.

B. Instruction Modelling Mitigates Overfitting of Instruction Tuning

This section explores the underlying interpretation behind the effectiveness of our approach. Our experimental results demonstrate that IM can alleviate the overfitting problem of Instruction Tuning. Below we will discuss our findings in detail.

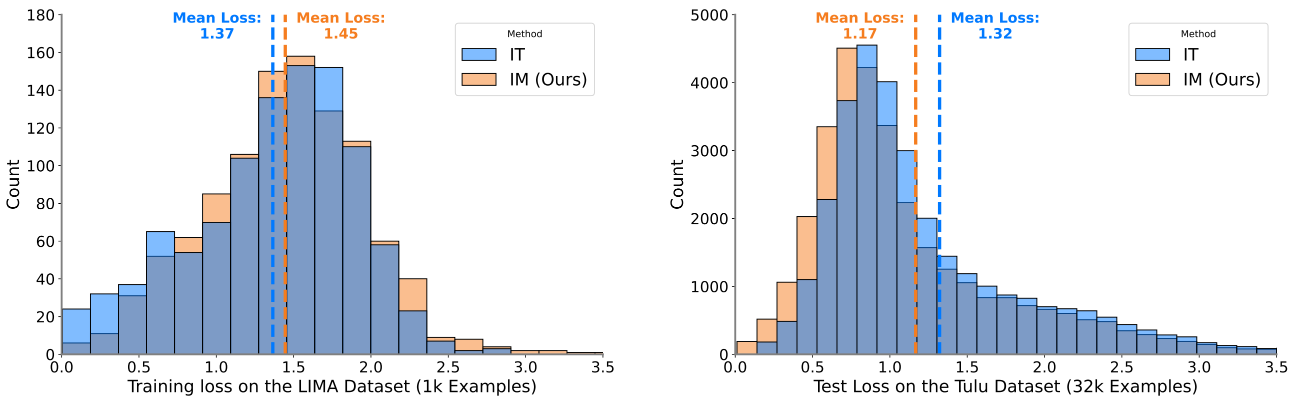

Figure 3. (**Left**) Training loss distribution for each example between our approach INSTRUCTION MODELLING (IM) and INSTRUCTION TUNING (IT) on the `LIMA` dataset. (**Right**) Test loss distribution for each example between IM and IT on the `Tulu V2` dataset, using a 10% randomly sampled data for efficacy. Mean losses are marked by dashed lines. For both IM and IT, here we only compute the loss over the output part. IM has a higher train loss with lower test loss, suggesting that IM effectively mitigates the overfitting issues compared to IT. See Appendix §G for more examples.

#1. Train and test loss analysis. Figure 3 clearly illustrates the effectiveness of our approach IM in mitigating overfitting issues compared to IT. In the training loss distribution for the `LIMA` dataset, IM exhibits a slightly higher mean loss of 1.45 compared to 1.37 for IT, suggesting that IM does not overfit to the training data as much as IT does. This is further corroborated in the test loss distribution on the `Tulu V2` dataset (using a 10% randomly sampled data set), where IM demonstrates a lower mean test loss of 1.17 compared to 1.32 for IT. This indicates that IM maintains better generalisation to new data, emphasising the model's capability to learn effectively without fitting excessively to training examples. For more examples, see Appendix §G.

Table 2. Average BLEU Score comparison of IM and IT, where a lower score indicates less overfitting. Green and red arrows indicate performance changes against the baseline (IT).

	`LIMA`	`Less Tydiqa`	`Less MMLU Chat`	`Less BBH ICL`	`Alpagasus Alpaca 5k`	`Alpagasus Dolly 9k`	`Alpagasus Dolly 3k`
IT	18.15	69.21	72.43	60.96	72.26	61.76	60.99
IM (ours)	17.30$\downarrow$0.85	65.63$\downarrow$3.58	69.20$\downarrow$3.23	53.94$\downarrow$7.02	70.50$\downarrow$1.76	60.61$\downarrow$1.15	59.04$\downarrow$1.95

#2. BLEU score analysis. Here we generate outputs using the instructions from the training examples via greedy decoding, and then compare the generated outputs with the ground truth outputs in training examples and report the results. We use BLEU (up to n-gram order 4) [40] to measure the similarity between outputs, where a higher score on outputs indicates a higher overlap with training examples. As shown in Table 2, outputs generated by IM consistently have lower BLEU scores than those generated by IT. This suggests that IM produces outputs have less overlap with the ground truth outputs in training examples, indicating less overfitting.

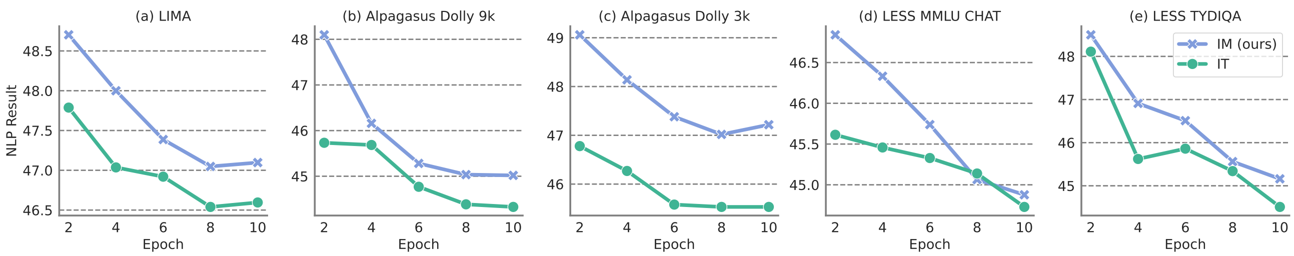

Figure 4. Mean performance on 18 NLP tasks over epochs using LLAMA-2-7B-BASE. This analysis suggests that IM experiences a lower instruction tuning tax compared to IT.

#3. Instruction Tuning Tax on the NLP tasks. Previous works show that training LMs with RLHF causes *Alignment Tax* on the NLP tasks [2, 38]. In this study, we observe that instruction tuning can sometimes lead to diminished model capabilities in some areas, such as multilinguality and commonsense reasoning. To this end, we further explore the impact of instruction tuning on the performance of NLP tasks. Figure 4 illustrates that our approach IM generally has a lower instruction tuning tax compared to IT over IT, suggesting better robustness under the low-resource setting. We provide additional experiments for win rates across epochs in Appendix §H.

Table 3. Performance on 18 NLP benchmarks and AlpacaEval 2.0. Green and red arrows indicate performance changes against the baseline (LLAMA-2-7B-BASE). This analysis suggests that while applying KL Loss in the instruction tuning helps mitigate performance degradation in NLP tasks, it substantially harms the model performance in open-ended generation tasks.

		LIMA (1K)		ALPAGASUS DOLLY (9K)	
	LLAMA-2-7B-BASE	**IT w/o KL Loss**	**IT w/ KL Loss**	**IT w/o KL Loss**	**IT w/ KL Loss**
NLP Tasks	49.32	48.79↓0.53	49.26↓0.06	45.54↓3.78	49.31↓0.01
AlpacaEval 2.0	0.01	2.58↑2.57	0.06↑0.05	2.28↑2.27	0.04↑0.03

#4. Can we use KL divergence loss as regularisation for instruction tuning? In this analysis, we explore the application of KL divergence loss in instruction tuning and assess its impact on both instruction following and model performance. Table 3 offers a detailed comparison across various NLP benchmarks and open-ended language generation tasks, particularly using AlpacaEval 2.0, with models trained with and without KL divergence loss. Our findings are as follows: (1) Incorporating KL Loss reduces overfitting and reduces the performance degradation on traditional NLP tasks. For example, on the `Dolly` dataset, incorporating KL Divergence Loss leads to less instruction tuning tax in NLP tasks, with scores rising from 45.54 to 49.31. (2) KL Loss detrimentally affects the model's instructions following abilities. For example, on the `LIMA` dataset, we observe a substantial decrease in AlpacaEval 2.0 scores from 2.58 to 0.06. For additional ablation studies and implementation details, see Appendix §I.

C. Further Analysis

#1. The advantage of our proposed method persists with different language models and sizes. As shown in Figure 5, our analysis demonstrates that our proposed method IM consistently outperforms the IT across different models and sizes, including OPT-6.7B and LLAMA-2-13B-BASE, on 18 traditional NLP tasks and AlpacaEval 1.0 benchmark These findings underline the effectiveness of our approach irrespective of the underlying language model or its scale.

#2. Relationship between the model output length and the win rate. As shown in Figure 6, win rates are not necessarily associated with the lengths of the outputs. Our result reveals that our approach IM does not necessarily generate longer outputs than IT across different data utilisation levels from the `Tulu V2` dataset. Specifically, the output lengths for

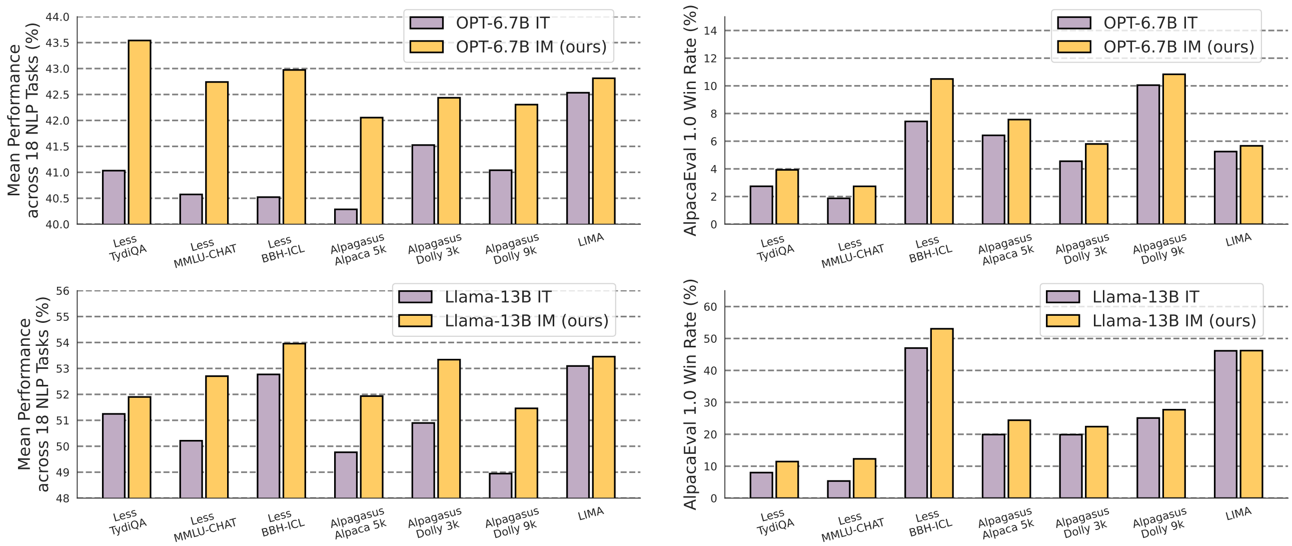

Figure 5. Comparison of INSTRUCTION TUNING (IT) and INSTRUCTION MODELLING (IM) methods using OPT-6.7B (**Top Row**) and LLAMA-2-13B-BASE (**Bottom Row**) trained on 7 instruction tuning datasets. (**Left**) The mean performance across 23 traditional NLP tasks. (**Right**) The win rate on the AlpacaEval 1.0 benchmark.

both approaches are similar despite varying levels of data utilisation. Furthermore, IM consistently outperforms the IT, suggesting that improvements in performance as measured by win rates on the AlpacaEval 1.0 are not dependent on the output length. We provide additional analysis on other instruction tuning datasets under the SAH in Appendix §J.

Table 4. Performance comparison of IM and IM +NEFTUNE on AlpacaEval 1.0 and various NLP benchmarks. Green and red arrows indicate performance changes against the baseline (IM). This analysis shows that adding NEFTUNE to IM could further improve model performance.

	`LIMA`	`Less Tydiqa`	`Less MMLU Chat`	`Less BBH ICL`	`Alpagasus Alpaca 5k`	`Alpagasus Dolly 9k`	`Alpagasus Dolly 3k`
AlpacaEval 1.0 Win Rate							
IM	32.94	10.10	9.78	44.15	19.52	30.77	15.11
IM +NEFTUNE	30.77↓2.17	23.41↑13.31	12.45↑2.67	48.25↑4.10	32.07↑12.55	38.28↑7.51	23.35↑8.24
Mean Performance Across 23 NLP Tasks							
IM	49.60	48.70	47.84	49.15	47.47	48.00	48.95
IM +NEFTUNE	49.47↓0.13	49.44↑0.74	47.73↓0.11	48.62↓0.53	48.70↑1.23	48.63↑0.63	49.54↑0.59

#3. Our proposed method IM could further improve the model performance with NEFTUNE. Table 4 demonstrates the combined effects of our proposed method IM and NEFTUNE on performance across various NLP tasks and the AlpacaEval 1.0 benchmark. The integration of NEFTUNE with IM generally further improves the win rates in AlpacaEval 1.0, showing notable improvements in several datasets such as a 13.31% increase on `Less Tydiqa` and a 12.55% boost on `Alpagasus Alpaca 5k` (in absolute). However, this combination leads to a performance drop in certain contexts, such as a lower performance on NLP tasks on `Less MMLU Chat` and `Less BBH ICL`. This indicates that while NEFTUNE may enhance model robustness under certain conditions, its benefits are context-dependent, highlighting the need for the careful application of NEFTUNE when used in conjunction with IM to optimise effectiveness across diverse evaluation settings.

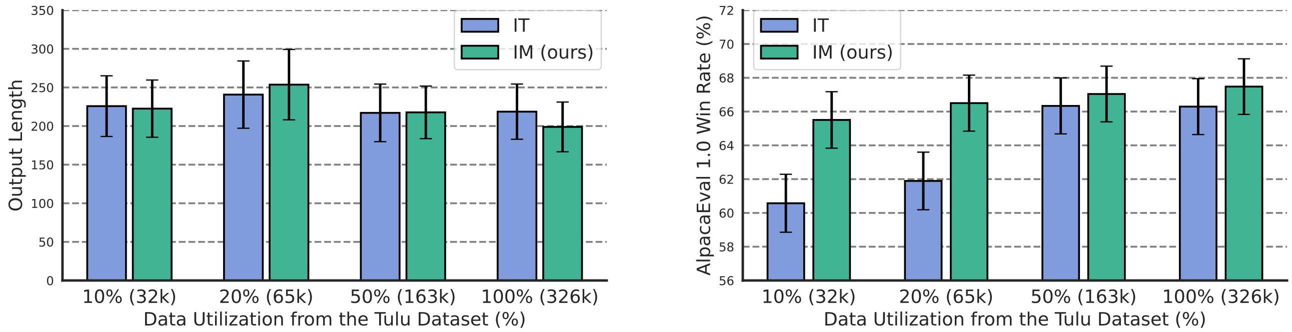

Figure 6. (**Left**) Output length comparison between our approach INSTRUCTION MODELLING (IM) and INSTRUCTION TUNING (IT) across various data utilisation levels from the `Tulu V2` dataset, as evaluated on the AlpacaEval dataset. (**Right**) Performance comparison (measured by win rate) between IM and IT on the AlpacaEval 1.0 across various data utilisation levels from the `Tulu V2` dataset. This analysis suggests that the improvement provided by IM is not necessarily associated with the increased output lengths. See more length analysis in Appendix §J.

D. Instruction Tuning Dataset

In this work, we use 13 popular datasets from previous instruction tuning research. For the `WizardLM`, `Sharegpt`, `Science Literature`, and `Code Alpaca` datasets, we directly use the subset provided by the previous work [24]. Refer to the dataset statistics in Table 5. In addition, we provide an analysis of the output length distribution for `LIMA`, `Alpagasus Dolly 3k`, `Alpagasus Dolly 9k`, `Alpagasus Alpaca 5k`, `Less MMLU Chat`, `Less Tydiqa`, and `Less BBH ICL` datasets, as shown in Figure 7.

Table 5. Statistical summary for various instruction tuning datasets. The table includes sample sizes, the average total length of instructions and outputs, the average output length, and the average instruction length with their standard deviations, and ratio calculations.

Dataset	Size	Total	Output	Output Std	Instruction	Instruction Std	Output/Instruction	Instruction/Output
`LIMA`	1,030	484.47	442.75	491.34	41.72	79.28	10.6124	0.0942
`Less MMLU Chat`	13,533	225.19	8.24	16.42	216.95	301.64	0.0380	26.3316
`Less Tydiqa`	13,533	172.44	25.13	42.62	147.31	235.37	0.1706	5.862
`Less BBH ICL`	13,533	262.03	61.44	92.55	200.60	196.79	0.3063	3.265
`Alpagasus Dolly 3k`	2,996	111.91	68.08	106.38	43.83	107.53	1.5530	0.6439
`Alpagasus Dolly 9k`	9,229	73.40	56.62	48.91	16.79	11.33	3.3727	0.2965
`Alpagasus Alpaca 5k`	5,305	48.29	30.81	34.44	17.48	12.45	1.7631	0.5672
`Tulu V2`	326,181	541.16	343.56	575.32	197.60	345.99	1.7387	0.5751
`Tulu V2` (10%)	32,618	517.45	338.96	562.74	178.49	345.72	1.8991	0.5266
`Tulu V2` (50%)	163,090	515.63	340.67	571.06	174.97	343.45	1.9470	0.5136
`Tulu V2` (20%)	65,236	504.56	336.89	562.46	167.68	331.24	2.0092	0.4977
`WizardLM`	30,000	350.05	258.35	182.98	91.71	86.09	2.8170	0.3550
`Sharegpt`	50,000	1035.39	831.15	757.10	204.24	344.51	4.0695	0.2457
`Science Literature`	7,544	1196.08	46.46	57.34	1149.62	905.99	0.0404	24.7417
`Stanford Alpaca`	52,002	63.77	45.18	44.97	18.59	12.42	2.4302	0.4115
`Code Alpaca`	20,022	49.74	27.40	27.35	22.34	10.67	1.2262	0.8156

E. Evaluation Datasets and Details

We use the open-source repositories, LM-Evaluation Harness[3] and Huggingface Dataset[4] as the evaluation tools. We describe our evaluation setup below:

[3] https://github.com/EleutherAI/lm-evaluation-harness
[4] https://huggingface.co/docs/datasets

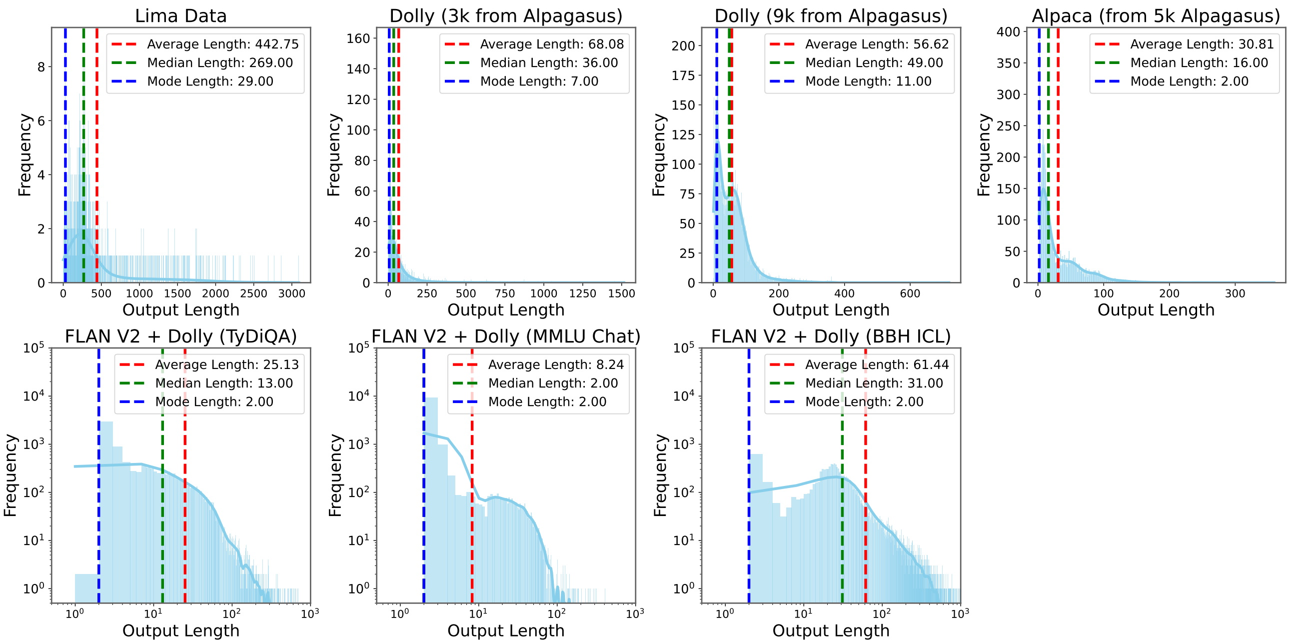

Figure 7. Distribution of output lengths of instruction tuning datasets. This figure presents histograms for the distribution of output lengths across seven datasets, including `LIMA`, `Alpagasus Dolly 3k`, `Alpagasus Dolly 9k`, `Alpagasus Alpaca 5k`, `Less MMLU Chat`, `Less Tydiqa`, and `Less BBH ICL`. Each subplot displays the frequency of output lengths with key statistical indicators: the average (red dashed line), median (green dashed line), and mode (blue dashed line) of each dataset. The last three subplots employ a logarithmic scale on both axes to better illustrate data spread.

MMLU. We evaluate the model using the dataset at the huggingface dataset [5]. We follow the protocol outlined in HuggingFace Open LLM Leaderboard [6]. The evaluation uses multiple-choice questions formatted as the question followed by four choices (A, B, C, D) and prompting for an answer. We calculate the mean accuracy (acc) across test examples.

BBH. The model evaluation utilizes the dataset at the huggingface dataset [7], specifically tested on the ‘test‘ split without the use of few-shot examples. We follow the setup in previous works [24, 48]. The evaluation metric is the exact match score, averaged (mean) to assess performance. Generation is constrained to a maximum of 1024 tokens, with termination upon encountering specific delimiters such as ”¡/s¿”, ”Q”, or double newlines. The generation is greedy decoding (temperature set to 0.0) and does not use sampling. Answer extraction employs regex patterns to identify responses immediately following ”the answer is” and captures only the first occurrence.

GSM8K. We evaluate using the dataset at the huggingface dataset [8], focusing on arithmetic problem-solving in the ‘test‘ split. We follow the HuggingFace Open LLM Leaderboard to 8 few-shot examples. Exact match is the chosen metric, with case insensitivity and select regex-based filtering of common punctuation and formatting characters to ensure precise validation of numerical answers. The primary focus is on extracting and comparing the final numerical answer to the model’s output using a strict regex-based match setup.

HumanEval. We evaluate using the dataset and the evaluation code from the previous work [24]. We report the performance of the pass@1. We perform the decoding using two different temperatures, 0.1 and 0.7. We report the better pass@1 from these two decoding results.

[5] `https://huggingface.co/datasets/hails/mmlu_no_train`
[6] `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`
[7] `https://huggingface.co/datasets/lukaemon/bbh`
[8] `https://huggingface.co/datasets/gsm8k`

ARC. The evaluation setup for the dataset at the huggingface dataset [9] utilizes a multiple-choice format. We follow the HuggingFace Open LLM Leaderboard to 25 few-shot examples. The performance metric used is mean normalized accuracy (acc_norm).

CoQA. We conduct the model evaluation on the dataset at the huggingface dataset [10]. We follow the HuggingFace Open LLM Leaderboard to 0 few-shot examples. The output generation terminates upon encountering a new line followed by "Q:". The mean F1 score is used as the evaluation metric.

PIQA. Evaluation on the dataset at the huggingface dataset [11] involves a multiple-choice. The evaluation incorporates 10 few-shot examples, according to the LIMIT [26]. Performance is measured using the mean normalized accuracy (acc_norm).

OpenBookQA. The dataset at the huggingface dataset [12] is evaluated in a multiple-choice format. The mean normalized accuracy (acc_norm) is used as the evaluation metric.

LAMBADA. The evaluation of the model on the dataset at the huggingface dataset [13] is performed using a loglikelihood output type. The mean accuracy is used as the evaluation metric.

HellaSwag. In the 'hellaswag' dataset at the huggingface dataset [14], model evaluation is conducted using a multiple-choice format. We follow the HuggingFace Open LLM Leaderboard to 10 few-shot examples. The mean normalized accuracy (acc_norm) is used as the evaluation metric.

The Winograd Schema Challenge. The evaluation is conducted using a multiple-choice format on the 'test' split at the huggingface dataset [15]. The mean accuracy is used as the evaluation metric.

Winogrande. The 'winogrande' dataset is assessed using a multiple-choice format at the huggingface dataset [16]. We follow the HuggingFace Open LLM Leaderboard to 5 few-shot examples. The mean accuracy is used as the evaluation metric.

LAMBADA. For this dataset, evaluation is conducted using the loglikelihood output type on the 'test' split at the huggingface dataset [17]. This variant focuses on predicting the last word of text passages in English. The mean accuracy is used as the evaluation metric.

Translation Benchmarks WMT. The evaluation of the translation capabilities is performed on the WMT 2014[18] and WMT 2016[19] datasets at the huggingface dataset. Here we use the 'ter' score as the evaluation metric.

TruthfulQA. We use the dataset at the huggingface dataset [20]. We follow the setup at the HuggingFace Open LLM Leaderboard using the 6 few-shot examples. The mean accuracy is used as the evaluation metric.

ToxiGen. We use the dataset at the huggingface dataset [21]. The task is assessed using a multiple-choice framework to evaluate the model's capability to identify hateful content in text statements. The mean accuracy is used as the evaluation metric.

[9] https://huggingface.co/datasets/allenai/ai2_arc
[10] https://huggingface.co/datasets/EleutherAI/coqa
[11] https://huggingface.co/datasets/piqa
[12] https://huggingface.co/datasets/openbookqa
[13] https://huggingface.co/datasets/lambada
[14] https://huggingface.co/datasets/hellaswag
[15] https://huggingface.co/datasets/winograd_wsc
[16] https://huggingface.co/datasets/winogrande
[17] https://huggingface.co/datasets/EleutherAI/lambada_openai
[18] https://huggingface.co/datasets/wmt14
[19] https://huggingface.co/datasets/wmt16
[20] https://huggingface.co/datasets/truthful_qa
[21] https://huggingface.co/datasets/skg/toxigen-data

Hendrycks Ethics. We use the dataset at the huggingface dataset [22], with a multiple-choice format. The model aims to detect whether described actions in various contexts are ethically wrong. The prompt format integrates a specific scenario followed by a structured question: ”Is this wrong?” and then prompts for an answer with options ’no’ or ’yes’. The mean accuracy is used as the evaluation metric.

F. Implementation Details

Experimental Design for Figure 2 Left. Here we present a detailed experimental design for Figure 2 Left. We perform experiments on a variety of datasets, including `LIMA`, `Alpagasus Dolly 3k`, `Alpagasus Dolly 9k`, `Alpagasus Alpaca 5k`, `Less MMLU Chat`, `Less Tydiqa`, `Less BBH ICL`, `Tulu V2`, `Code Alpaca`, `Stanford Alpaca`, `Science Literature`, `WizardLM`, and `Sharegpt`. Furthermore, to evaluate the effectiveness of IM on datasets with different instruction-to-output length ratios, we select three subsets from `Tulu V2`. Each subset contains 3,000 training examples, with instruction-to-output length ratios of approximately 5, 10, and 15, respectively.

Experimental Design for Figure 2 Right. Here we provide a detailed experimental design for Figure 2 Right. We strategically sampled varying sizes of training examples from the `Tulu V2` dataset to investigate the effectiveness of IM with different sizes training examples. Starting with approximately 320,000 examples in the `Tulu V2` dataset, we creates subsets of data ranging from as few as 1,000 to as many as 35,000 examples. These subsets were selected randomly, ensuring a representative mix across different scales. We adhered to a fixed instruction-to-output length ratio of approximately 10 to maintain consistency in training conditions across all samples. We train the LLAMA-2-7B-BASE on all these subsets and evaluate them respectively.

Table 6. Hyperparameters and configurations for supervised fine-tuning.

Hyperparameter	Assignment
GPUs	2 or 4 A100 80G GPUs
Batch size per GPU	1
Total batch size	128
Number of epochs	2, 3, or 10
Maximum sequence length	2048
Learning rate	2×10^{-5}
Learning rate optimizer	AdamW
Adam epsilon	1e-6
Adam beta weights	0.9, 0.98
Learning rate scheduler	Linear with warmup
Warmup proportion	0.03
Weight decay	0
Mixed precision	bf16
Gradient accumulation steps	Calculated dynamically

Implementation Details. In our study, we fine-tune the LLaMA-2-7B, LLaMA-2-13B and OPT-6.7 model using four A100 80G GPUs, with a per-GPU batch size of 1 and a total batch size of 128, employing a learning rate of 2e-5. Training typically proceeds for 2 epochs with a maximum sequence length of 2048 tokens. We utilise gradient accumulation, calculated to effectively distribute training steps across the available hardware, resulting in larger batch sizes despite hardware limitations. We employ mixed precision (bf16), linear learning rate scheduling with a warm-up ratio of 0.03, and a

[22] https://huggingface.co/datasets/EleutherAI/hendrycks_ethics

weight decay of 0. To optimise our training, we use DeepSpeed with a stage 3 configuration without offloading. Our setup also includes the usage of Flash Attention [14] and slow tokenization to enhance training efficiency and compatibility. Our code is implemented using Open-Instruct[23], Pytorch[24] and Huggingface[25]. Table 6 lists the hyperparameters.

G. Train and Test Loss

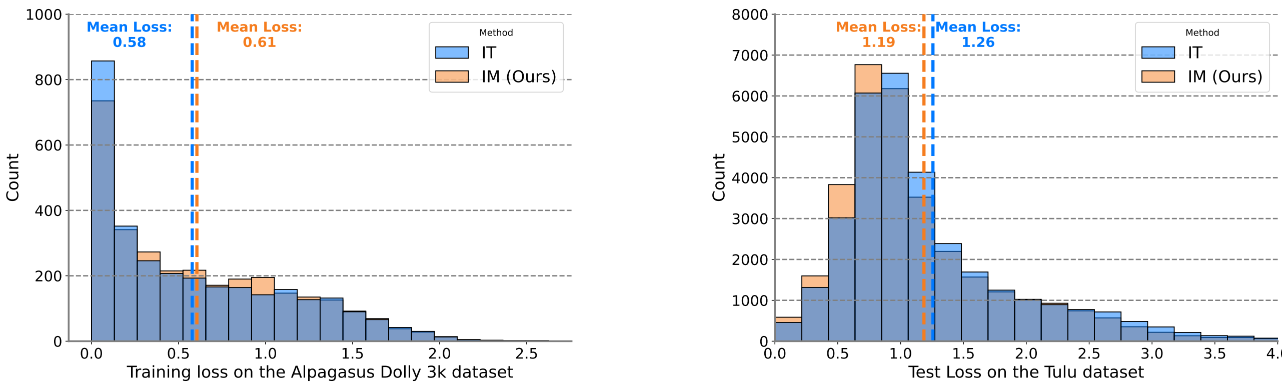

Figure 8. (**Left**) Training loss distribution for each example between our approach INSTRUCTION MODELLING (IM) and INSTRUCTION TUNING (IT) on the `Alpagasus Dolly 3k` dataset. (**Right**) Test loss distribution for each example between IM and IT on the `Tulu V2` dataset, using a 10% sampled data. Mean losses are marked by dashed lines. For both IM and IT, here we only compute the loss over the output part. IM has a higher train loss with lower test loss, suggesting that IM effectively mitigates the overfitting issues compared to IT.

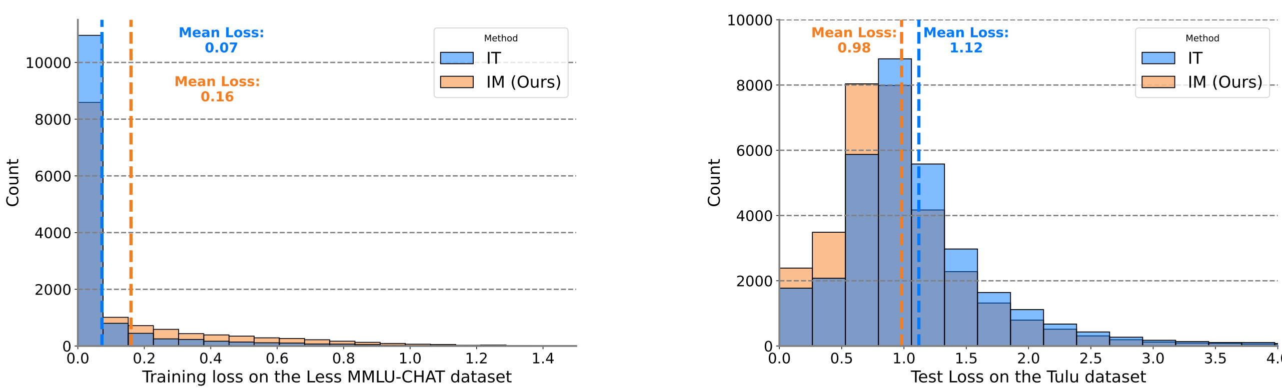

Figure 9. (**Left**) Training loss distribution for each example between our approach INSTRUCTION MODELLING (IM) and INSTRUCTION TUNING (IT) on the `Less MMLU Chat` dataset. (**Right**) Test loss distribution for each example between IM and IT on the `Tulu V2` dataset, using a 10% sampled data. Mean losses are marked by dashed lines. For both IM and IT, here we only compute the loss over the output part. IM has a higher train loss with lower test loss, suggesting that IM effectively mitigates the overfitting issues compared to IT.

In this section, we provide additional experiments regarding training and testing loss distributions. Figure 8 focuses on the `Alpagasus Dolly 3k` and `Tulu V2` datasets, displaying how IM tends to exhibit higher training losses yet achieves lower test losses compared to IT. Similarly, Figure 9 compares these methods on the `Less MMLU Chat` and `Tulu V2` datasets under analogous conditions.

[23] `https://github.com/allenai/open-instruct`
[24] `https://pytorch.org/`
[25] `https://huggingface.co/`

H. The impact of Epochs on the Win Rate

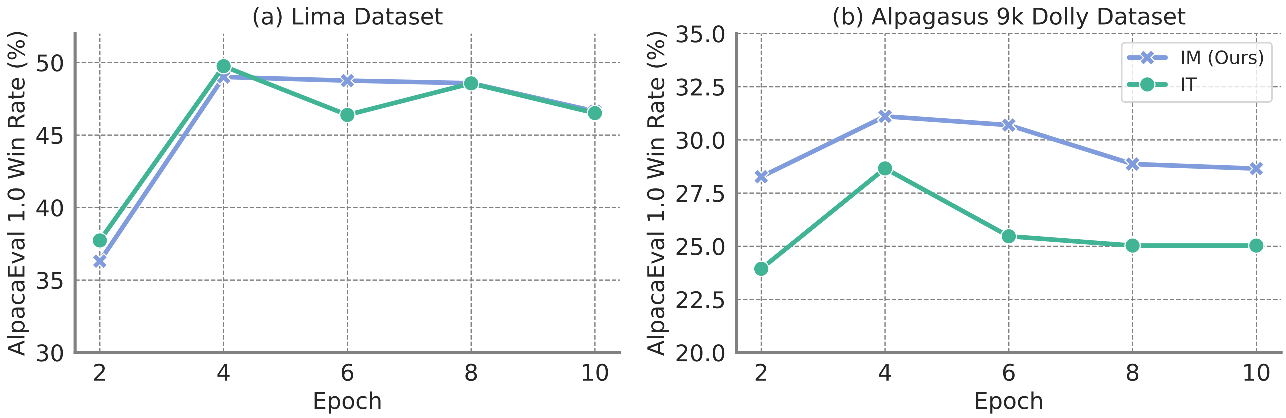

Figure 10. AlpacaEval 1.0 performance trends for IM and IT approaches on the `LIMA` and `Alpagasus Dolly 9k` datasets across different epochs.

The figure 10 illustrates the comparative analysis of AlpacaEval 1.0 scores across different epochs for two datasets, `LIMA` and `Alpagasus Dolly 9k` datasets. We evaluate the performance of IM and IT over different numbers of epochs. IM consistently surpasses IT in performance on the `Alpagasus Dolly 9k` dataset, while the performance of both approaches is comparable on the `LIMA` dataset.

I. Applying KL Divergence Loss for Instruction Tuning

In this section, we first briefly introduce the Kullback-Leibler (KL) divergence, and then introduce the experimental details.

Kullback-Leibler Divergence. Kullback-Leibler (KL) divergence is commonly employed as a regularisation method in the fine-tuning of LMs, helping to mitigate overfitting by constraining the fine-tuned model to remain similar to the pre-trained model [38]. Specifically, the KL divergence is added to the fine-tuning objective as a per-token regularisation term between the fine-tuned LM $\pi_\theta(x)$, and the pre-trained LM, $\pi^{\text{pre}}(x)$. For supervised fine-tuning with next token prediction loss, the training objective incorporating KL divergence is computed as follows:

$$\mathcal{L}_{\text{KL}}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\big[\sum_t -\log \pi_\theta(x_t|x_{0:t-1}) + \lambda \sum_t \text{KL}(\pi_\theta(x_t|x_{0:t-1})||\pi^{\text{pre}}(x_t|x_{0:t-1}))\big], \quad (2)$$

where λ is a regularisation parameter that balances the loss due to the next token prediction and the KL divergence, and $\pi(x_t|x_{0:t-1})$ represents the next token distribution of the fine-tuned or pre-trained LM conditioned on the preceding context.

Table 7. Performance on 23 NLP benchmarks and AlpacaEval 2.0, with various values of λ, trained on the (LLAMA-2-7B-BASE).

	NLP Tasks	**AlpacaEval 2.0**
LLAMA-2-7B-BASE	49.32	0.01
$\lambda = 0.01$	48.81	2.58
$\lambda = 0.1$	48.77	2.44
$\lambda = 1.0$	49.26	0.06

Ablation study on the effect of λ. In Table 3, we set the value of the λ as 1.0. Here we provide additional experiments with different values of λ. Table 7 presents the model performance on the NLP tasks and AlpacaEval 2.0. This aligns our observations in §B.

J. The impact of Epochs on Output Lengths

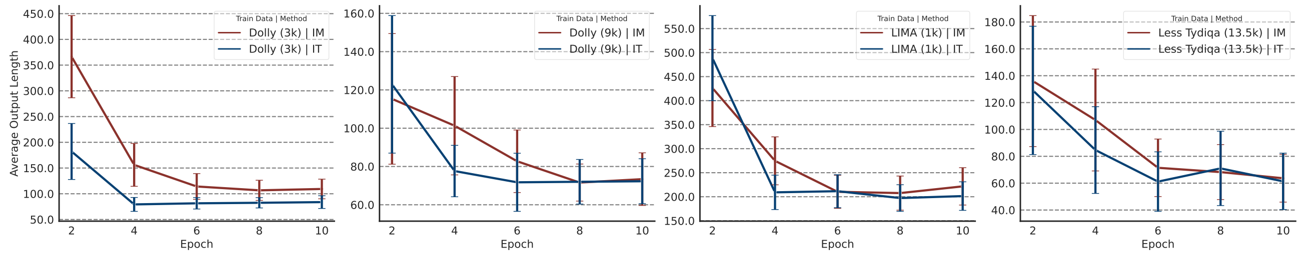

Figure 11. Comparative analysis of output lengths for IM and IT across different epochs on `Alpagasus Dolly 3k`, `Alpagasus Dolly 9k`, `LIMA`, and `Less Tydiqa` datasets.

Figure 11 illustrate the average output length of various models across different epochs. We report the output length on four different datasets, including `Alpagasus Dolly 3k`, `Alpagasus Dolly 9k`, `LIMA`, and `Less Tydiqa`. Each line represents the average output length of a model, with epochs ranging from 2 to 10, and is accompanied by error bars that denote the normalised standard deviation (10%) of the output lengths.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one.

The \onecolumn command above can be kept in place if you prefer a one-column appendix, or can be removed if you prefer a two-column appendix. Apart from this possible change, the style (font size, spacing, margins, page numbering, etc.) should be kept the same as the main body.