Improving Multimodal Large Language Models Using Continual Learning

Shikhar Srivastava[1]*, Md Yousuf Harun[2], Robik Shrestha[1], Christopher Kanan[1]

[1]University of Rochester, [2]Rochester Institute of Technology

Abstract

Generative large language models (LLMs) exhibit impressive capabilities, which can be further augmented by integrating a pre-trained vision model into the original LLM to create a multimodal LLM (MLLM). However, this integration often significantly decreases performance on natural language understanding and generation tasks, compared to the original LLM. This study investigates this issue using the LLaVA MLLM, treating the integration as a continual learning problem. We evaluate five continual learning methods to mitigate forgetting and identify a technique that enhances visual understanding while minimizing linguistic performance loss. Our approach reduces linguistic performance degradation by up to 15% over the LLaVA recipe, while maintaining high multimodal accuracy. We also demonstrate the robustness of our method through continual learning on a sequence of vision-language tasks, effectively preserving linguistic skills while acquiring new multimodal capabilities.

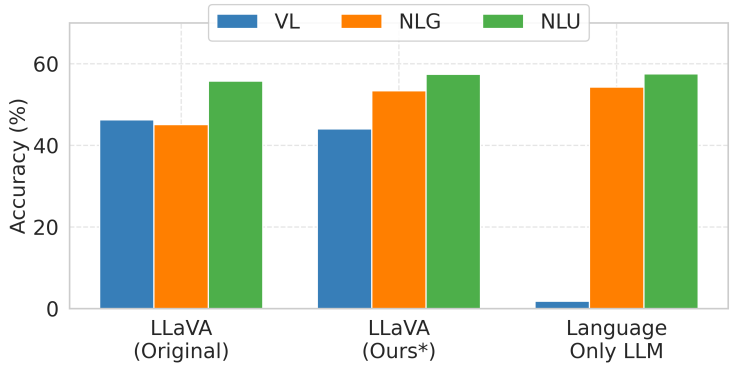

Figure 1: Summary results of the best CL methods we evaluated for training LLaVA 1.5 compared to the unimodal base LLM and the original version of LLaVA 1.5. All results are with Pythia 2.8B as the base LLM. The best method has almost the same vision-language (VL) accuracy while providing a large increase in linguistic performance on 1 NLG and 4 NLU tasks by 8% and 2% (absolute), resp.

1 Introduction

Advances in integrating visual information with large language models (LLMs) have led to the development of multimodal large language models (MLLMs), excelling at many vision-language (VL) tasks [1–12]. Recent studies converge on a general recipe for developing MLLMs: Alignment of LLM token embeddings with visual embeddings followed by instruction-tuning on VL tasks like visual question answering (VQA) [8]. However, creating an MLLM often degrades the LLM's natural language understanding (NLU) and generation (NLG) performance, a phenomenon known as

*Corresponding author: shikhar.srivastava@rochester.edu

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

catastrophic forgetting [3, 6]. For instance, PaLM-E experienced an 87% drop in NLG performance over the base LLM [6]. Similar forgetting has been noted for LLaVA [11], but little work has addressed understanding and mitigating this issue. Multimodal LLMs are designed in part to serve as general multimodal understanding models [13, 14]. They must therefore perform well not only on vision-language data, but also retain their linguistic or text-only performance.

Here, we study mitigating the loss of linguistic abilities in the popular LLaVA MLLM using continual learning (CL) techniques designed to mitigate catastrophic forgetting [15]. In CL, a sequence of non-stationary tasks is learned, where we treat the first task as already learned by the base LLM followed by new VL tasks. We study these methods in two paradigms. In the first, we seek to recreate LLaVA 1.5 while mitigating linguistic forgetting through our methods, and in the second, we sequentially learn each VL dataset in the LLaVA recipe.

This paper makes the following contributions:

1. Using the original LLaVA 1.5 training recipe, we study linguistic forgetting in 9 MLLMs, including 5 built on the Pythia family of LLMs to study the role of model scales and instruction tuning on such linguistic forgetting.
2. We study the effectiveness of 5 mitigation techniques for reducing linguistic forgetting and show that the best method improves accuracy for NLG, NLU, and VL tasks compared to the naive LLaVA recipe (see Fig. 1).
3. We pioneer studying CL for MLLMs by sequentially learning VL tasks, where we assess the efficacy of CL techniques to mitigate catastrophic forgetting in this challenging scenario.

2 Methods

2.1 The LLaVA MLLM

We study LLaVA 1.5, henceforth referred to as LLaVA, which is one of the most widely used multi-modal training protocols. LLaVA has the following components:

1. **Visual Encoder:** Following earlier implementations [16, 12], we use a pre-trained ViT-L/14 from CLIP which takes an image resolution of 336px as the vision encoder, which is kept frozen throughout training to ensure stability, prevent overfitting to initial training tasks.
2. **LLM:** We study 9 choices for LLaVA's LLM of varying scales and instruct-tuning: 6 Pythia models (160M - 2.8B) [17], Phi2 (3B) [18], and 2 LLaMA 2 (7B) models. Refer to Appendix A.3 for details on the base LLMs.
3. **Alignment Network:** To inject other modalities into LLaVA, it uses a two layered alignment network that projects embeddings from the vision encoder into the embedding representational space of the text tokens [19].

We follow the standard LLaVA 1.5 training recipe, and provide detailed implementational descriptions in Appendix A.3.

2.2 Continual Learning Methods

To mitigate catastrophic forgetting in MLLMs, we examine and adapt several continual learning methods for LLaVA multimodal training: LoRA [20], Soft Targets [21], Rehearsal [22], and mSGM [21], along with the original LLaVA fine-tuning (Naive FT). Details of the mitigation methods, the continual learning setup and data mixtures are provided in Appendix A.

In all methods, the alignment layer and LLM are trained and the ViT is frozen. Following the LLaVA 1.5 training protocol, the VL datasets are trained in a single epoch corresponding to a single training pass through each dataset.

3 Experiments

In our experiments, we treat training a *MLLM as a CL problem*, where the system learns a sequence of tasks from 1 to T. Task 1 always consists of training the LLM, where we assume the LLM has already been trained and we do not know the provenance of the training data or the exact methods to create it, which is true of many commonly used LLMs (e.g., Llama 2 and Llama 3). In our experiments on analyzing and mitigating linguistic forgetting, there are 2 tasks: 1) learning the base LLM, and 2) learning the mixture of VL datasets. In CL experiments, there are 5 tasks where the first is training the base LLM, and then each VL task is sequentially learned.

Following the standard LLaVA recipe, for all experiments, task 2 begins by training the VL alignment network using the LLaVA-595 CC-SBU captioning dataset. The network is trained to generate captions with an auto-regressive loss, with the vision encoder and LLM kept frozen. Subsequently, the LLM and alignment layer are trained for tasks 2 to T. For the forgetting and mitigation experiments in Secs. 3.1 and 3.2, the two-task setup is identical to LLaVA 1.5's training protocol. The continual learning setup employed in Sec. 3.3 is detailed in Appendices A and B.

Evaluation. We evaluate the models using six natural language datasets: Lambada [23] for NLG, ARC-Easy [24], ARC-Challenge [24], Winogrande [25], and WSC [25] for NLU, as well as four vision-language datasets: VQAv2 [26], GQA [27], TextVQA OCR and Pure [28], and RefCOCO [29] corresponding to the LLaVA data mixture (see Appendix B.3 for details). Performance is measured using accuracy and the forgetting metric Δ, which quantifies changes in task performance following training on new tasks (see Appendix A.3).

3.1 Analyzing Linguistic Forgetting

Linguistic forgetting has not been studied in LLaVA, so before assessing CL methods, we measure linguistic forgetting and VL performance for all LLMs using the standard LLaVA training recipe; which uses naive fine-tuning. Using this recipe, our 9 LLMs are transformed into MLLMs.

Overall Results. Our overall results are given in Fig. 2a. Five models suffer from linguistic forgetting, while surprisingly, four have increased NLU/NLG accuracy due to positive transfer from the VL tasks.

Positive Backwards Transfer & Analysis of NLU vs. NLG Tasks. In Table 2b , we study the impact of forgetting on NLU vs. NLG tasks. All MLLMs exhibit greater NLG forgetting than NLU forgetting, and NLU datasets are the source of the cases of positive backward transfer (negative Δ). We posit this may be due to the additional common-sense reasoning and world knowledge encoded in visual-language tasks and instructions, which is relevant for the NLU tasks. In terms of model size, typically smaller models show higher NLG forgetting, which is consistent with PaLM-E's observations. We include additional results on NLU, NLG forgetting in Appendix Sec. C.3.

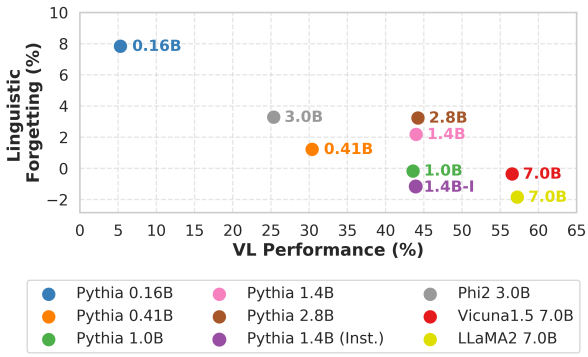

Figure 2(a): Linguistic forgetting versus VL perf. for 9 MLLMs trained with the LLaVA recipe. Five models exhibited linguistic forgetting, while four had negative linguistic forgetting, indicating that VL training resulted in positive transfer to language tasks.

LLM	Scale (B)	NLU $\Delta\downarrow$	NLG $\Delta\downarrow$
Pythia	0.16	0.94	12.01
	0.41	–1.19	8.62
	1.0	–1.63	4.95
	1.4	0.55	8.07
	2.8	1.74	9.18
Pythia (I)	1.4	–1.20	–1.01
Phi2	3.0	2.60	4.39
Vicuna 1.5	7.0	–0.98	2.04
LLaMA 2	7.0	–2.15	–0.43
Average	—	–0.15	5.31

Figure 2(b): **NLU vs NLG Forgetting:** Composition of linguistic forgetting between NLU and NLG tasks for models after LLaVA training. Negative Δ indicates positive backward-transfer, which is desirable. "I" denotes the instruct-tuned model.

3.2 Mitigating Linguistic Forgetting

We study the efficacy of our mitigation methods toward reducing linguistic forgetting. Due to computational constraints, we exhaustively tested mitigation methods with Pythia (160M) in Table 3a, and then evaluate the best method across all parameter scales of Pythia models in Fig. 3b.

In Table 3a, Soft Targets has the highest accuracy across VL datasets with the least linguistic forgetting. LoRA and mSGM better preserve NLU/NLG performance but at the cost of decreased VL accuracy compared to naive fine-tuning. We posit that the *retention of next-token logits* enforced by Soft Targets is crucial in mitigating linguistic forgetting for causal generation during MLLM training.

Model	Vision-Language (VL) ↑ VQAv2	TextVQA OCR	TextVQA Pure	GQA	VL Avg. Acc ↑	NL Avg. Δ ↓	NL Avg. Acc ↑
Pythia (160M)	0.00	0.00	0.00	0.00	0.00	–	**32.61**
Naive FT	30.32	2.40	3.83	22.17	5.29	7.83	24.78
LoRA	28.97	1.02	1.74	17.97	2.42	**1.69**	30.92
mSGM	28.39	1.37	2.71	17.48	3.36	2.68	29.93
Soft Targets	**32.67**	**6.92**	**6.10**	**25.39**	**10.57**	2.83	29.78

Figure 3(a): **Results for mitigation methods on LLaVA training.** Evaluate alternatives to naive fine-tuning for transforming Pythia 160M into an MLLM. In Fig. 3b (**right**), we evaluate the average linguistic forgetting for Pythia models in the 160M to 2.8B scale, after training on LLaVA. Negative forgetting refers to a *positive backward transfer* in NL performance after multimodal training.

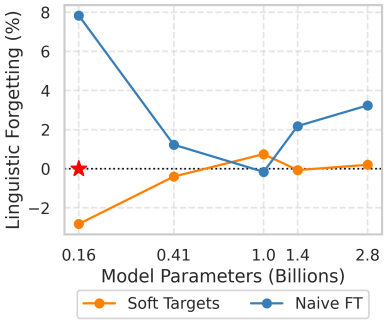

Figure 3(b): Linguistic forgetting for varying model sizes of Pythia (160M - 2.8B).

Analyzing the Role of Parameter Count. In Driess et al. [6], larger models had less catastrophic forgetting than smaller ones for NLG/NLU. The opposite result was found in Luo et al. [30].
We analyze this phenomenon in the Pythia family of models in Fig. 3b, where we evaluate naive fine-tuning and the best method for 160M, Soft Targets. Across scales, Soft Targets has zero or negative linguistic forgetting over the naive fine-tuning method used in the original LLaVA paper, with competitive VL performance compared to naive fine-tuning (especially at higher scales, see Fig. 4a). Soft Targets achieved positive backward transfer in the low model size regime (0.16-0.41B) and no forgetting for larger sizes (>0.41B). In contrast, naive fine-tuning had severe forgetting in the 0.16-0.41B regime and reduced forgetting when the model size exceeded 1B parameters.

3.3 Continually Learning VL Tasks

We next turn to continually learning each of the VL datasets used to train LLaVA 1.5, where we group the datasets based on the task type. Details of the data mixture, task groupings and ordering are provided in Table 4 and Sec. B.3 in the Appendix. Note that there is no VL evaluation dataset associated with the Task 2 (Instruct), but we still measure NLU/NLG performance. Given our limited computational budget, we exhaustively studied our CL methods only for the smaller-scale Pythia 410M LLM (see Table 1), and then we evaluated the best-performing mitigation method for all of the Pythia LLMs (see Appendix Figures 5a and 5b).

Table 1: **Continually learning LLaVA Tasks with Pythia 410M.** We report *cumulative* task-wise accuracy and forgetting of each mitigation method across VL and NL tasks, where we evaluate test sets associated with all tasks seen up to current task. Task 5 represents cumulative performance.

Model	Task 2 (Instruct) VL (A ↑)	Task 2 (Instruct) NL (Δ ↓)	Task 3 (VQA) VL (A ↑)	Task 3 (VQA) NL (Δ ↓)	Task 4 (OCR) VL (A ↑)	Task 4 (OCR) NL (Δ ↓)	Task 5 (Ref) VL (A ↑)	Task 5 (Ref) NL (Δ ↓)
Naive-FT	-	**0.58**	**44.22**	12.21	**16.67**	4.95	0.48	7.70
Soft Targets	-	0.81	0.16	14.67	10.23	5.41	0.31	10.90
LoRA	-	1.38	37.46	1.23	14.03	2.50	9.59	4.36
mSGM	-	1.11	36.31	1.57	11.69	**1.40**	0.32	6.32
Rehearsal (1%)	-	**0.58**	37.74	10.90	3.47	7.41	3.55	7.65
mSGM + Reh. (1%)	-	1.11	35.28	**0.73**	12.38	2.25	**10.21**	**2.77**

Results for Pythia 410M. Results for Pythia 410M are given in Table 1. In terms of NLU/NLG forgetting, all mitigation methods showed efficacy in reducing forgetting across the sequence of tasks. Overall, mSGM with and without rehearsal achieves the least linguistic forgetting while maintaining the highest *cumulative* VL performance across all baselines. Regarding VL performance, naive fine-tuning achieves better performance at the cost of high linguistic forgetting. In contrast, mSGM achieves competitive VL performance and sometimes even surpasses it, e.g., for RefCOCO, an especially challenging VL task.
Model scaling results. Model scaling results are given in Appendix Fig. 5. We compare mSGM with Rehearsal - the best method identified for Pythia 410M, against naive LLaVA fine-tuning across all Pythia model sizes. In general, the original naive LLaVA fine-tuning exhibits greater loss of NLU/NLG as well as VL performance. In contrast, VL performance for mSGM rivals or exceeds the original LLaVA fine-tuning across all model sizes, with little to no NLU/NLG forgetting.

4 Discussion

This work presents one of the first studies of linguistic forgetting in MLLMs, particularly for open-source models with modest parameter counts (< 7B). We show that the degree of linguistic forgetting typically reduces with model scale, and is far more severe for NLG tasks compared to NLU. We pioneer treating MLLM creation as a CL problem, and show that CL methods effectively mitigate linguistic forgetting while minimally hindering VL accuracy. In our experiments, our best mitigation method far outperforms the naive LLaVA recipe in terms of linguistic forgetting, while maintaining competitive VL performance. We show that this benefit holds across model scales. We pioneer CL for MLLMs, and establish strong baselines for this task. In fact, we observe that besides maintaining competitive VL performance with naive LLaVA training, our mitigation methods achieve near zero and in some cases even negative linguistic forgetting. This suggest that our mitigation methods achieve a positive backward transfer of linguistic ability after multi-modal training. Given the essential nature of multimodal abilities for many applications, our research highlights that naive multimodal fine-tuning can significantly degrade prior linguistic abilities. Our findings underscore the need to develop more robust and capable mitigation approaches, and showcase the applicability of CL techniques to the fine-tuning of foundation models. We aim to inspire future research in this direction, ultimately contributing to the advancement of more resilient and versatile MLLMs.

Acknowledgments

This work was supported in part by NSF awards #2326491 and #2317706. The views and conclusions contained herein are those of the authors and should not be interpreted as representing any sponsor's official policies or endorsements.

References

[1] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

[2] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

[3] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35: 23716–23736, 2022.

[5] 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, March 2023. URL `https://lmsys.org/blog/2023-03-30-vicuna/`.

[6] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. *ICML*, 2023.

[7] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models. *arXiv preprint arXiv:2304.10592*, 2023.

[8] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[9] Gen Luo, Yiyi Zhou, Tianhe Ren, Shengxin Chen, Xiaoshuai Sun, and Rongrong Ji. Cheap and quick: Efficient vision-language instruction tuning for large language models. *arXiv preprint arXiv:2305.15023*, 2023.

[10] Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. Grounding language models to images for multimodal inputs and outputs. *ICML*, 2023.

[11] Ji Lin, Hongxu Yin, Wei Ping, Yao Lu, Pavlo Molchanov, Andrew Tao, Huizi Mao, Jan Kautz, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. *arXiv preprint arXiv:2312.07533*, 2023.

[12] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023.

[13] Ji Lin, Hongxu Yin, Wei Ping, Pavlo Molchanov, Mohammad Shoeybi, and Song Han. Vila: On pre-training for visual language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26689–26699, 2024.

[14] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, et al. Mm1: Methods, analysis & insights from multimodal llm pre-training. *arXiv preprint arXiv:2403.09611*, 2024.

[15] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[16] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36, 2024.

[17] Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR, 2023.

[18] 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 arXiv:2306.11644*, 2023.

[19] Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. Linearly mapping from image to text space. *arXiv preprint arXiv:2209.15162*, 2022.

[20] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[21] Md Yousuf Harun and Christopher Kanan. Overcoming the stability gap in continual learning. *arXiv preprint arXiv:2306.01904*, 2023.

[22] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017.

[23] Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

[24] 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 arXiv:1803.05457*, 2018.

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

[26] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6904–6913, 2017.

[27] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019.

[28] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.

[29] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 787–798, 2014.

[30] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[31] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. *arXiv preprint arXiv:2402.07865*, 2024.

[32] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

[33] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[34] Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. Textbooks are all you need ii: phi-1.5 technical report. *arXiv preprint arXiv:2309.05463*, 2023.

[35] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[36] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

[37] 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 arXiv:2307.09288*, 2023.

[38] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[39] Damjan Kalajdzievski. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*, 2023.

[40] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

Appendix

We organize implementation details and additional supporting experimental findings as follows:

- Appendix A describes the implementation details for the experiments.
- Appendix B provides details on the datasets and evaluations.
- Appendix C summarizes findings of additional supporting experiments and ablation studies.

A Implementation Details

We provide details below of the experimental settings for the alignment and fine-tuning phases of the LLaVA1.5 training protocol, the continual learning methods methods we employ, and the implementational and hyperparameter search details. For LLaVA, our configuration details are the same as LLaVA1.5 in order to reproduce and compare effectively to the LLaVA1.5 multi-modal training protocol.

A.1 Libraries and Tools

To reproduce the LLaVA 1.5 experiment, we build on top of the Prismatic library [31], and use the Prismatic library for vision-language tasks and EleutherAI's LM_Eval library [32] for natural language evaluations. All our code is written in PyTorch [33].

A.2 Base LLMs used

We study 9 choices for LLaVA's LLM: comprising six Pythia models [17], Phi2 (3B)[18], and two LLaMA 2 (7B) models. The six Pythia variants span various scales, including 160M, 410M, 1B, 1.4B, and 2.8B parameters, with two versions of the 1.4B model—one of which is instruction fine-tuned. Additionally, we evaluate both the original LLaMA 2 and the instruction fine-tuned Vicuna-1.5 7B[5], which is utilized in LLaVA 1.5.

1. **Phi2** [34] has been trained on the same dataset as Phi1 [18], which includes a curated selection of "textbook quality" data from the web (6 billion tokens) and an additional 1 billion tokens of synthetically generated textbooks and exercises created using GPT-3.5 [35]. The Phi series are among the most performative models in the under 7B parameter size class [34].
2. **Pythia** [17] comprises two sets of 8 models, each corresponding to two datasets. For every model size, one set is trained on the Pile dataset [36], while the other set is trained on a version of the Pile where global de-duplication has been applied. The granular model scaling suite of Pythia is particularly useful for our study. We select the de-duplicated set of Pythia models.
3. **LLaMA 2** is reportedly trained on a mix of publicly available online data, but specific details are not available [37]. LLaVA 1.5 reports its best performance with instruction fined-tuned LLaMA 2 LLMs [12].

A.3 LLaVA Training Details

Following the LLaVA 1.5 protocol [8], the visual encoder is a CLIP VIT-L@336px which takes an image resolution of 336px with letterbox resizing, and the alignment network is a two-hidden layer MLP projector with GELU activation. The LLM generations are limited to a maximum token length of 2048.

A.3.1 Alignment Stage

In the alignment stage, we use a learning rate of 0.001 with a linear warmup followed by a cosine decay scheduler. The training process is carried out for 1 epoch with a global batch size of 256, distributed as 16 samples per device. We use gradient checkpointing and mixed precision training to speed up computation. We used FSDP (Fully Sharded Data Parallel) to train all our models [38]. During the alignment stage, we shard only the gradients and optimizer states, not the parameters.

A.3.2 Fine-tuning Stage

For the fine-tuning stage, the learning rate is $2e-05$, with a linear warmup for a 0.03 ratio of steps followed by cosine decay. This stage also runs for 1 epoch but with a smaller global batch size of

128, keeping the per-device batch size constant at 16. Depending on the size of the LLM (Phi2 3B vs Pythia 160M), we vary the per-device batch size, but the global batch size is kept constant across all experiments. We switch to the FSDP full sharding strategy, with all parameters, optimizer states, and gradients sharded across the devices.

A.4 Continual LLaVA

For the Continual LLaVA setting, we follow the same experimental configuration as before in the LLaVA setting. A model trained on Task (i), will then be simply used as the pretrained checkpoint for Task $(i+1)$, with training following normally. The training configurations are then identical to the fine-tuning stage. In the case of PET methods like LoRA, the adapter weights are merged back into the LLM before starting training on the new task.

Different from the LLaVA setting, we continually learn each of the VL datasets used to train LLaVA 1.5, where we group the datasets based on the task type. Details of the dataset sequence are described in Appendix 4.

A.5 Continual Learning Methods

To mitigate catastrophic forgetting in MLLMs, we examine and adapt several methods for LLaVA multimodal training:

1. **Naive Fine-Tuning** corresponds to the original LLaVA method with no modifications.
2. **LoRA** keeps the original LLM weights frozen and learns low-rank updates for them [20]. Following LoRA's recommended protocol, we inject LoRA weights into all LLM linear layers. After each task is learned, LoRA weights are merged into the LLM. Details in Appendix A.6.2.
3. **Soft Targets** was proposed in [21] to reduce forgetting in CL. Rather than using hard targets for training, simple label smoothing is used to reduce the severity of the training loss under distribution shift in CL. We smooth the hard target vector Y, by smoothing the target tokens by $-\alpha$, and offsetting non-target tokens by $+\alpha/(N-1)$, where α controls the smoothing, and N is the LLM's vocabulary size. We discuss the choice of α in Appendix A.6.3.
4. **Rehearsal (Experience Replay)** is an effective method for CL that involves storing data from earlier tasks and mixing it with data from new tasks. We study it in our CL experiments since we do not have any stored data for task 1 (training the base LLM). We study storing 1% of randomly selected samples from each previous task, excluding task 1.
5. **mSGM** is based on SGM, which combines soft targets, weight initialization, and LoRA to mitigate catastrophic forgetting [21]. We adapt SGM based on our modified soft targets and omit weight initialization and output layer freezing since the output vocabulary is static, and the output layers are used for the causal generation of LLMs.

A.6 Hyperparameter Search

A.6.1 LLaVA1.5 Setting

To reproduce and directly compare against the LLaVA 1.5 protocol, we keep the explicit training configurations the same (as mentioned in Appendix A).

A.6.2 LoRA

We train several different LoRA settings on the smallest Pythia-160M LLM for tractability and compare the resulting VL and NLU/NLG performances. We vary the a) target modules (between 1) all linear layers, and 2) key, query, and value projection layers only), b) LoRA ranks in the range of $1/4 - 1/2$ of the full rank of the model, rank stabilized LoRA [39], and larger alpha values (16 instead of 8 as default). Table 2 shows the comparisons by training on the LLaVA setting.

A.6.3 Soft Targets

We train the LLaVA recipe with the soft targets with varying alpha $\alpha \in \{0.001, 0.01, 0.1\}$, and report results in Figure 3. The Pythia 160M model is used for this tuning.

Table 2: **Analysis of LoRA Ranks and Configuration:** We train the Pythia 160M model with a varying set of ranks and configurations.

Model	Vision-Language (VL)				VL Avg.	NL Avg.	
	VQAv2	TextVQA OCR	TextVQA Pure	GQA	Acc ↑	Δ ↓	Acc ↑
Original LLaVA	30.32	2.40	3.83	22.17	5.29	7.83	24.78
Language Only LLM	0.00	0.00	0.00	0.00	0.00	-	32.61
LoRA (1/2 Full Rank, Higher Alpha)	28.72	1.05	2.67	19.73	2.84	9.33	23.28
LoRA (1/2 Full Rank, RSLoRA)	28.97	1.02	1.74	17.97	2.42	1.69	30.92
LoRA (1/4 Full Rank)	24.64	0.93	1.41	15.04	2.11	11.64	20.97
LoRA (1/4 Full Rank, Higher Alpha)	6.46	0.68	0.55	2.44	1.04	2.53	30.08
LoRA (1/2 Full Rank)	0.13	0.20	0.10	0.00	0.00	-	-
LoRA (1/2 Full Rank, RSLoRA, KQV Target)	0.00	0.00	0.00	0.00	0.00	-	-

Table 3: **Selecting α for Soft Targets**. We train the Pythia 160M model with Soft Targets by varying the $alpha \in \{0.001, 0.01, 0.1\}$.

Model	Vision-Language (VL)				VL Avg.	NL Avg.	
	VQAv2	TextVQA OCR	TextVQA Pure	GQA	Acc ↑	Δ ↓	Acc ↑
Language Only LLM	0.00	0.00	0.00	0.00	0.00	-	32.61
Soft Targets ($\alpha = 0.1$)	3.38	1.19	1.32	1.76	1.62	0.67	31.95
Soft Targets ($\alpha = 0.01$)	32.67	6.92	6.10	25.39	10.57	2.83	29.78
Soft Targets ($\alpha = 0.001$)	25.12	2.17	1.73	14.84	3.49	4.97	27.64
Original LLaVA	30.32	2.40	3.83	22.17	5.29	7.83	24.78

B Datasets & Evaluation

B.1 Training Datasets

For the continual LLaVA setting, the original LLaVA 1.5 data mixure is split into groups of vision-language (VL) tasks. These VL tasks are then learned sequentially in the Continual LLaVA training. Table 4 provides the splits of the LLaVA 1.5 data mixture into a sequence of VL tasks, based on task types. The collation of all these datasets forms the LLaVA 1.5 data mixture, which is used to train the LLaVA MLLMs, per the LLaVA's protocol as described in Appendix A.3.

Table 4: **Continual LLaVA Setup**: The LLaVA 1.5 data mixture is split into groups of vision-language (VL) tasks. VQA (OE & OK) refers to open-ended and outside-knowledge VQA tasks.

Task Type	Task	Data	Size
Pre-Training	1	LLM Pre-Training*	–
Instruct Tuning	2	CC-LAION-SBU	558K
		LLaVA-Inst, ShareGPT	198K
VQA (OE & OK)	3	VQA2	83K
		OKVQA	9K
		A-OKVQA	66K
		GQA	72K
VQA (OCR)	4	OCRVQA	80K
		TextCaps	22K
Referential Grounding	5	RefCOCO	48K
		VisualGenome	86K

B.2 Evaluation Datasets

Natural Language Evaluation: We use six datasets. For NLG, we use Lambada [23], which is the only NLG dataset that uses accuracy for evaluation. For NLU, we use ARC-Easy [24], ARC-Challenge [24], Winogrande [25], and WSC [25] for NLU.

Vision-Language Evaluation: We use the test sets corresponding to each VL dataset used for training LLaVA: VQAv2 and GQA for general VQA tasks [26, 27], TextVQA OCR (and Pure) for OCR tasks [28], and RefCOCO for referential expression generation tasks [29]. Additional details are given in Appendix B.3.

B.3 Evaluation Dataset Preparation

Corresponding to the datasets used to train LLaVA, we evaluate the model on all corresponding datasets. We use the *slim*[2] versions of all VQAv2, GQA, Text-VQA and RefCOCO datasets for evaluation, as provided within Prismatic-VLMs [31]. All *slim* versions of the evaluation sets contain 1024 examples each, and we use the provided index splits for testing.

B.4 Measuring Performance

To assess performance, we compute the forgetting metric Δ, where a positive value indicates forgetting and a negative value indicates learning the next task enhances the performance of previously acquired tasks (or a backward transfer [40]). The performance change, Δ, on task t after training on task k is defined as:

$$\Delta_t(k) = \omega_t(1) - \omega_t(k), \quad \forall t < T \tag{1}$$

where T is the total number of tasks, and ω_t is the harmonic mean of the accuracy values of the evaluation datasets for task t. We use the harmonic mean because it is dominated by the accuracy of the *worst-performing* dataset for the task, thereby emphasizing the need to perform well and avoid forgetting for all of the datasets.

C Additional Experiments

C.1 LLaVA Model Scaling

Below, we report the VL performance results with varying model scales after LLaVA training. We also again show the linguistic forgetting here for reference. In Figure 4a, we see that the VL performance of the Soft Targets approach is competitive with the Naive FT approach, especially at the 2.8B scale. We note that the gap in VL performance reduces with the model scale in the LLaVA, and is nearly on par with Naive FT at the highest scales. This is while maintaining a zero to negative loss in linguistic abilities compared to the Naive FT approach (Figure 4b).

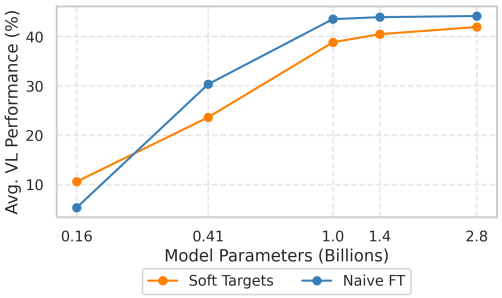

Figure 4(a): Avg. VL performance with model scale

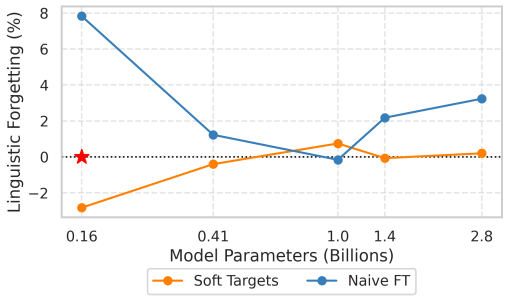

Figure 4(b): Linguistic forgetting with model scale

Figure 4: **Vision-language and linguistic forgetting for varying model sizes**. We evaluate the average VL performance and linguistic forgetting for Pythia models in the 160M to 2.8B scale, after training on LLaVA. Negative forgetting refers to a positive backward transfer in NL performance after multimodal training.

C.2 Continual LLaVA Model Scaling

Model scaling results are given in Fig. 5. We compare mSGM with Rehearsal, the best method identified for Pythia 410M with naive fine-tuning. Naive fine-tuning leads to a sharp and consistent drop in NLU/NLG performance as continual multi-modal training proceeds. In contrast, mSGM with Rehearsal has little to no NLU/NLG forgetting across all tasks and multiple model scales. For Pythia 1B, mSGM with Rehearsal achieves positive backward transfer for NLU/NLG datasets as the VL tasks are learned, which is rarely observed in CL. For the 160M parameter model, naive fine-tuning suffers large amounts of forgetting compared to mSGM. For some tasks, naive fine-tuning exhibits greater losses of NLU/NLG performance, unlike mSGM. In general, VL performance for mSGM rivals or exceeds naive fine-tuning across scales.

[2] https://github.com/TRI-ML/vlm-evaluation

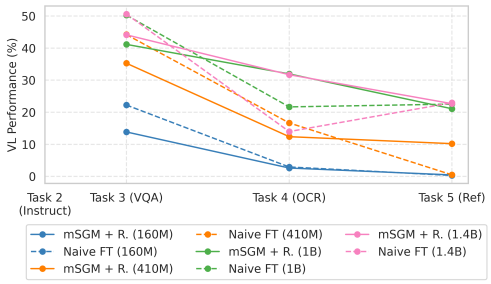

Figure 5(a): Continual VL Performance

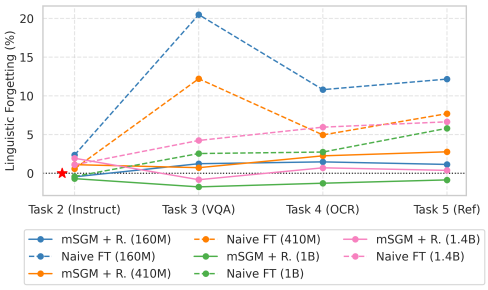

Figure 5(b): Continual Linguistic Forgetting

Figure 5: **Continual Learning LLaVA Tasks: Vision-language performance and Linguistic forgetting for varying model sizes**. We evaluate mSGM + Rehearsal (1%) and LLaVA Naive-FT on the Continual Learning setup, with varying base LLMs: Pythia models from 160M to 1.4B scale. Task 2 is not associated with any VL dataset for evaluation, since it is a captioning and instruction following task.

C.3 Analysis of Forgetting across NLU and NLG tasks

To understand the composition of linguistic forgetting on the LLaVA setting, we look at the NLU and NLG forgetting for 9 different LLMs with varying scales: Pythia scaling family of models from the 160M to the 2.8B parameter range, Phi2 3B, Vicuna1.5 7B and LLaMA2 7B. Figure 6 shows these results. We can observe a clear trend of higher forgetting for the NLG dataset (Lambada), compared to forgetting for NLU. Another trend we note is that forgetting typically reduces with higher model scales, for both NLG and NLU.

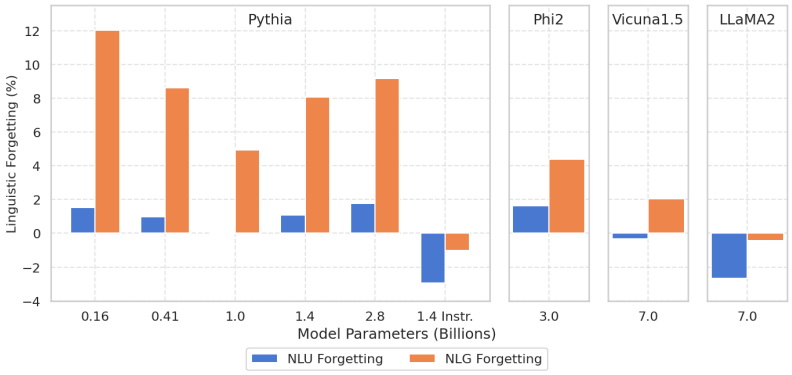

Figure 6: **Linguistic forgetting by NLU and NLG tasks**: For the LLaVA setting, we look at different model scales and families, and show the degree of linguistic forgetting by both NLU and NLG tasks separately. Here, the NLU and NLG averages are computed as the simple mean.

C.4 Pre-trained LLMs

Here we provide clickable links to download each of the open-source pre-trained LLMs used in this paper:

- phi-2-3b
- pythia-160m
- pythia-410m
- pythia-1b
- pythia-1p4b
- pythia-1p4b-instruct
- pythia-2p8b
- llama2-7b-pure
- vicuna-v15-7b

C.5 Artifact Use

We ensure that any artifacts (such as datasets, software, models, code, or other supplementary materials) associated with our paper are used in a manner that aligns with their original purpose and the guidelines set forth by the creators. In particular, our artifacts are the models and code we build our experiments on top of, which we have listed above in Sections A.1 and C.4.