ECE1512 Project B Report
Part 2 - LLaVA

Rémi Grzeczkowicz
MScAC Student
University of Toronto - Department of Computer Science
Student Number: 1010905399
remigrz@cs.toronto.edu

For this part, we choose to work on LLaVA designed by [7].

I Summary of the LLaVA model

LLaVA is a model designed to extend the capabilities of large language models (LLMs) to handle vision-and-language tasks. It achieves this by integrating a vision encoder that processes images into a suitable representation. This image encoding is then concatenated with the textual input and provided as input to the LLM.

A. LLaVA key ideas

The key ideas of LLaVA are:

- **Multimodal Instruction-Following Data**: LLaVA addresses the challenge of limited vision-language instruction-following data by leveraging GPT-4 to convert image-text pairs into an instruction-following format. This approach generates diverse and high-quality multimodal instruction data, which is essential for effectively training the model.
- **Large Multimodal Model Architecture**: The model integrates the CLIP visual encoder [10] with the Vicuna language model [2]. This combination enables the model to perform a variety of tasks, including classification, detection, segmentation, and captioning, in an open-world context. The architecture aligns visual features with language embeddings, facilitating seamless interaction between modalities.
- **Instruction Tuning**: LLaVA employs a two-stage instruction-tuning process. First, it pre-trains on a large dataset to align visual and language features. Then, it fine-tunes on task-specific datasets, such as ScienceQA [9], to improve its reasoning capabilities and enhance its ability to follow instructions across various tasks.
- **Evaluation Benchmarks**: To assess its performance, LLaVA introduces two benchmarks: LLaVA-Bench (COCO) [6], which evaluates consistency on structured visual inputs, and LLaVA-Bench (In-the-Wild) [1], which challenges the model on diverse and complex tasks. These benchmarks help measure the model's instruction-following capabilities and its generalization to novel domains.
- **Open Source Contributions**: The project provides open access to its codebase, model checkpoints, and generated multimodal instruction data, fostering further research and development in multimodal AI systems.

B. Technical Contributions

TThe technical contributions of LLaVA are:

- **Introduction of Multimodal Instruction-Following Data**: The authors developed a new pipeline for generating multimodal instruction-following datasets. They transformed image-text pairs into instruction-following formats using symbolic representations (e.g., captions and bounding boxes) and GPT-4. This pipeline produced 158K multimodal samples, including: 58K conversation-based Q&A, 23K detailed descriptions, and 77K complex reasoning tasks. This innovation addressed the scarcity of high-quality multimodal instruction-following data.
- **Development of Large Multimodal Model Design**: The authors integrated CLIP's visual encoder with Vicuna, an open-source language decoder. They employed a lightweight linear projection layer to map visual features into the language embedding space, enabling efficient alignment. The architecture of the model is illustrated in Figure 1.
- **Training Approach**: The authors introduced a two-stage tuning process. The first stage involves pre-training on CC3M-derived data to align visual and language features, balancing concept coverage with training efficiency. The second stage focuses on fine-tuning using the generated multimodal datasets for two distinct applications: (1) as a multimodal chatbot, and (2) for scientific Q&A tasks.
- **Benchmark Design for Multimodal Instruction Following (LLaVA-Bench)**: The authors designed a benchmark to evaluate LLaVA as a multimodal chatbot. This benchmark includes two components: (1) a COCO-based benchmark

to test consistency in visual and language instruction alignment using diverse visual content, and (2) an In-the-Wild benchmark to evaluate performance on challenging real-world multimodal tasks, such as memes, sketches, and paintings.
- **State-of-the-Art Performance on ScienceQA**: LLaVA demonstrated state-of-the-art performance on the ScienceQA dataset, which requires reasoning over scientific text and images. This highlights the model's ability to generalize to novel domains and perform complex multimodal reasoning tasks.

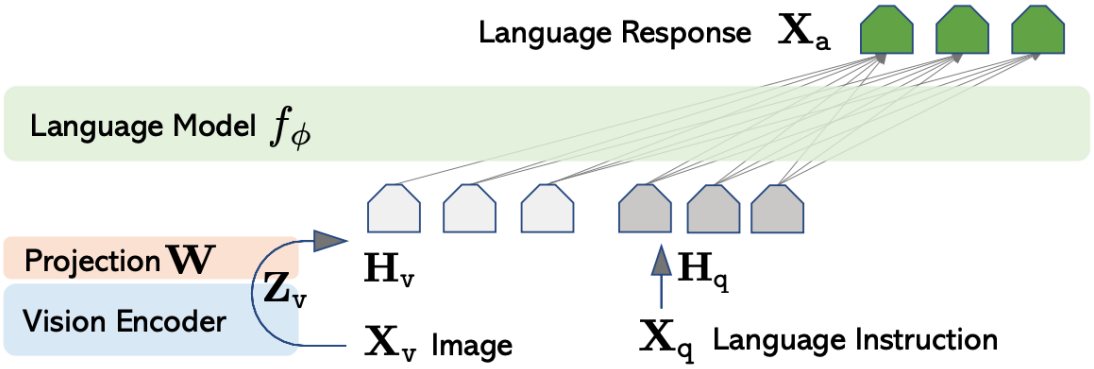

Fig. 1: LLaVA architecture

C. Areas for Improvement

The authors suggest several areas for improvement:

- **Hallucination and Bias**: Similar to other large language models (LLMs), LLaVA may produce hallucinated outputs. This issue is likely inherited from the underlying LLM.
- **Bias**: LLaVA may inherit biases from either the LLM or the visual encoder. These biases could manifest in the generated instructions or the model's predictions, affecting fairness and reliability.
- **Evaluation**: The authors acknowledge that evaluating a model like LLaVA is challenging. They recommend extending evaluations to cover more diverse tasks and datasets, including specific tests for hallucinations.
- **Visual Encoder**: The authors chose to freeze the visual encoder during the fine-tuning stage. However, training the visual encoder could potentially improve the model's performance. This could be achieved using techniques such as Low-Rank Adaptation (LoRA) [4].
- **Sophisticated Model Architectures**: Currently, LLaVA employs a simple linear layer to project visual features into the language embedding space. The authors suggest exploring more advanced architectures, such as gated cross-attention mechanisms, transformers, or other sophisticated integration techniques, to better align visual and language data and enhance overall performance.

II Efficiency improvements

According to [11], the LLM is the primary factor contributing to computational cost, as the visual encoder is typically small and the layer projecting Z_V to H_V is a simple linear layer. Consequently, the number of tokens fed to the LLM is the main determinant of computation cost.

The computational cost of an LLM arises from the number of parameters in the model. The larger the model, the slower its performance. However, as noted in [13], larger models often deliver better performance. This presents a trade-off: reducing the model size can improve speed but may also decrease accuracy.

To address this challenge, the authors of [11] propose a method to reduce the number of visual tokens while maintaining the model's performance (see Figure 2).

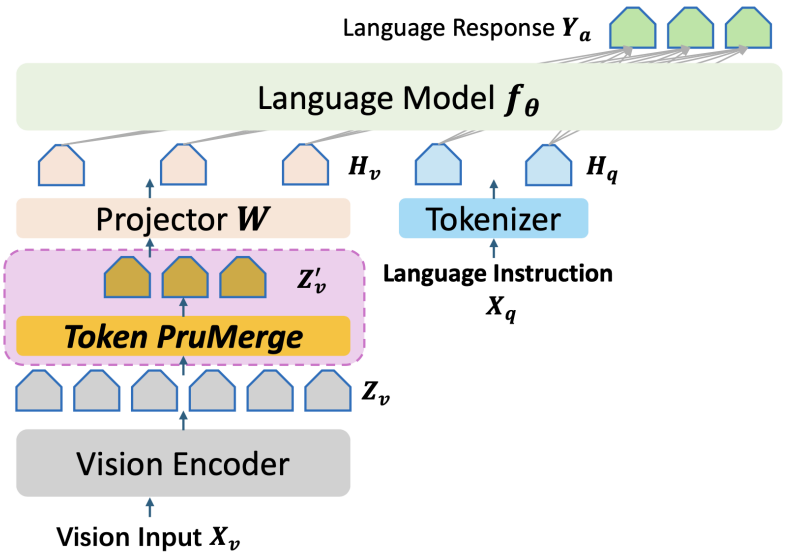

Fig. 2: Schema of LLaVA-PruMerge [11]

A. *LLaVA-PruMerge*

To reduce the number of visual tokens, the authors of [11] propose a method to merge visual tokens. The idea is to merge similar visual tokens and remove outliers. The process consists of the following steps:

- **Class-Visual Attention**: First, the authors compute the attention between the class token and the visual tokens using the following formula:

$$a_{cls} = \text{softmax}\left(\frac{q_{cls}K^T}{\sqrt{d_k}}\right) \tag{1}$$

  Here, q_{cls} represents the query vector of the class token, K is the key matrix of the visual tokens, and d_k is the dimensionality of the key vectors.
- **Outlier Removal**: The authors remove outlier visual tokens using the Interquartile Range (IQR) method. Specifically, they calculate the difference between the 75th percentile and the 25th percentile of the attention weights. Visual tokens with attention weights more than 1.5 times the IQR below the 25th percentile or above the 75th percentile are identified as outliers and removed.
- **Merging**: The authors merge similar visual tokens using a k-nearest neighbors (k-NN) algorithm. For each selected token, they find the k most similar visual tokens (including both selected and unselected tokens). They then compute a weighted sum of these tokens to create a new, merged token.
- **Token Supplement**: To address potential over-aggressiveness in token removal, the authors propose a token supplement step. They identify similarities between tokens using their "key" vectors (from the transformer's self-attention mechanism). Pruned tokens are grouped with the most relevant retained tokens, which act as cluster centers. These pruned tokens are then merged into the clusters through a weighted averaging process, enriching the retained tokens with additional information.

The algorithm for LLaVA-PruMerge is presented in Algorithm 1, and its architecture is illustrated in Figure 3.

Algorithm 1: Token PRUMERGE algorithm for reducing the number of visual tokens. [11]

Input: Key and Query matrices of ViT's penultimate layer, $\mathbf{K} = \{\mathbf{k}_1, \cdots, \mathbf{k}_n\}$ and $\mathbf{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_n\}$. The penultimate layer's output tokens, $\mathbf{Y} = \{\mathbf{y}_1, \cdots, \mathbf{y}_n\}$, where n is the number of input visual tokens.

Output: Refined $\mathbf{Y}$ to m (adaptive) visual tokens $\mathbf{Y}' = \{\mathbf{y}'_1, \cdots, \mathbf{y}'_m\}$, where $m \ll n$.

Token PRUMERGE:

1) Calculate attention between visual token and class token $a_{[\text{cls}]}$ using Equation 1.
2) Use the outlier detection algorithm IQR to adaptively select m important visual tokens' indices $\{i_1, \cdots, i_m\}$ based on $a_{[\text{cls}]}$.
3) **for** $p \in \{i_1, \cdots, i_m\}$ **do**
   Calculate the distance between selected token $\mathbf{y}_p$ and other visual tokens, $\mathbf{y}_{\{1,\cdots,n\}\setminus p}$;
   Use $\mathbf{y}_p$ as the cluster center and run k-nearest neighbor algorithm to find k similar tokens with indices $\{j_1, \cdots, j_k\}_p$;
   Update cluster center token with weighted sum: $\mathbf{y}'_p = \sum_{q=1}^{k} a[j_q] \cdot \mathbf{y}_{j_q}$;
   end
4) Token supplement: find the k-nearest neighbors of the unpruned tokens and merge them back into the clusters using their similarity score as the metric where the similarity score is calculated using the dot product of the key vectors.
5) Output a refined stack of visual tokens $\mathbf{Y}' = \{\mathbf{y}'_1, \cdots, \mathbf{y}'_m\}$.

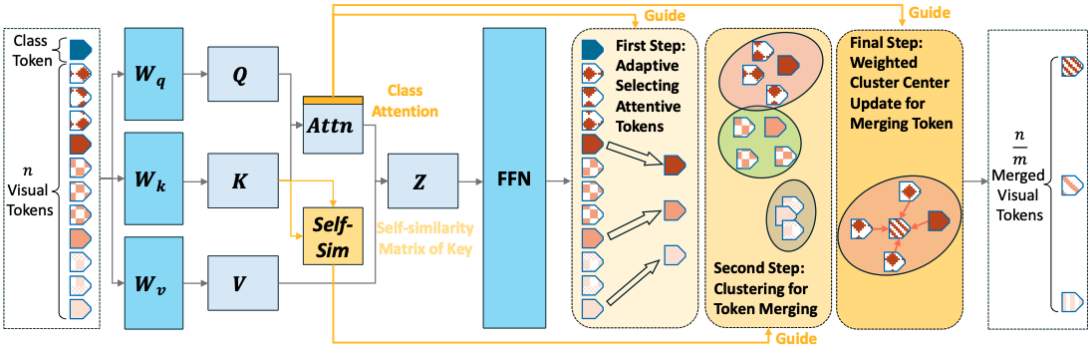

Fig. 3: LLaVA-PruMerge architecture [11]

B. Experimentation and results

To measure the efficiency of LLaVA-PruMerge, we planned to submit a set of images and prompts to both LLaVA and LLaVA-PruMerge and compare the time taken to generate outputs. Additionally, we intended to compare the outputs of both models to evaluate if their performance was similar.

For a fair comparison, we chose to use the 7B versions of both models. Since LLaVA-PruMerge models are only available with LoRA, we ensured consistency by using LoRA for both models.

Unfortunately, due to technical issues, we were unable to set up LLaVA (and consequently LLaVA-PruMerge) on our local machines. Despite extensive efforts on both macOS and Linux systems with GPU-enabled machines over several days, the setup was unsuccessful.

As a result, we rely on the experimental results reported in [11]. In their experiments, the authors compared the performance of LLaVA and LLaVA-PruMerge (along with other models) in their default versions and fine-tuned versions (trained for one epoch using LoRA) across multiple benchmarks. The results are presented in Table I.

Model	VQAv2 [3]	ScienceQA [9]	TextVQA [12]	POPE [5]	MME [14]	MMBench [8]
LLaVA	78.5	66.8	58.4	85.9	1510.7	60.9
LLaVA-PruMerge	72.0	68.5	56.0	86.3	1350.3	60.9
LLaVA (fine-tuned)	80.0	71.6	61.3	85.9	1531.3	67.7
LLaVA-PruMerge (fine-tuned)	72.8	71.0	58.4	86.2	1428.2	62.3

TABLE I: Results of LLaVA and LLaVA-PruMerge on different benchmarks [11]

We observe that LLaVA-PruMerge achieves performance comparable to LLaVA on most benchmarks. Notably, it outperforms LLaVA on ScienceQA and POPE benchmarks.

In terms of efficiency, the results are summarized in Table II.

Method	LLM Backbone	Quantization	FLOPs (T)
LLAVA-1.5	Vicuna-7B	FP16	9.3
LLAVA-1.5 w/ PruMerge	Vicuna-7B	FP16	0.91
LLAVA-1.5	Vicuna-7B	INT4	2.3
LLAVA-1.5 w/ PruMerge	Vicuna-7B	INT4	0.28
LLAVA-1.5	Vicuna-13B	FP16	18.2
LLAVA-1.5 w/ PruMerge	Vicuna-13B	FP16	1.80
LLAVA-1.5	Vicuna-13B	INT4	4.6
LLAVA-1.5 w/ PruMerge	Vicuna-13B	INT4	0.45

TABLE II: Efficiency of LLaVA and LLaVA-PruMerge on different benchmarks [11]

We observe that LLaVA-PruMerge is significantly more efficient than LLaVA. The number of FLOPs is reduced by approximately a factor of 10 when using LLaVA-PruMerge, representing a substantial improvement in efficiency.

III Conclusion

In this report, we presented the LLaVA model and its efficiency enhancement, LLaVA-PruMerge. We discussed the key concepts behind LLaVA, its technical contributions, areas for improvement, and the LLaVA-PruMerge method. We then presented the results of LLaVA-PruMerge across various benchmarks and compared its efficiency to LLaVA. We found that LLaVA-PruMerge offers much greater efficiency than LLaVA while maintaining comparable performance. This represents a significant advancement in efficiency.

References

[1] GitHub - Computer-Vision-in-the-Wild/CVinW_Readings: A collection of papers on the topic of "Computer Vision in the Wild (CVinW)" — github.com. https://github.com/Computer-Vision-in-the-Wild/CVinW_Readings. [Accessed 02-12-2024].

[2] 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.

[3] 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.

[4] 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.

[5] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv preprint arXiv:2305.10355*, 2023.

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[7] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023.

[8] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, pages 216–233. Springer, 2025.

[9] Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in Neural Information Processing Systems*, 35:2507–2521, 2022.

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[11] Yuzhang Shang, Mu Cai, Bingxin Xu, Yong Jae Lee, and Yan Yan. Llava-prumerge: Adaptive token reduction for efficient large multimodal models. *arXiv preprint arXiv:2403.15388*, 2024.

[12] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8317–8326, 2019.

[13] Chuhan Wu and Ruiming Tang. Performance law of large language models. *arXiv preprint arXiv:2408.09895*, 2024.

[14] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. A survey on multimodal large language models. *National Science Review*, page nwae403, 2024.

Appendix

The plagiarism detection score is approximately 26%. Upon reviewing the report, it appears that the majority of the detected plagiarism comes from the references section. Therefore, excluding the references, the plagiarism score is around 13%, which is acceptable. I hereby declare that this report is my own work and that I have not used any sources other than those cited in the references section.