Appendix
FOLDER: Accelerating Multi-modal Large Language Models with Enhanced Performance

Haicheng Wang[1,2,4*], Zhemeng Yu[1,2*], Gabriele Spadaro[2,3], Chen Ju[4], Victor Quétu[2], Shuai Xiao[4✉], Enzo Tartaglione[2✉]
[1] Paris Elite Institute of Technology, Shanghai Jiao Tong University [3] University of Turin
[2] LTCI, Télécom Paris, Institut Polytechnique de Paris [4] Taobao & Tmall Group of Alibaba
anakin_skywalker@sjtu.edu.cn, shuai.xsh@alibaba-inc.com, enzo.tartaglione@telecom-paris.fr

In this appendix, we first provide more details about our FOLDER module and experiment setup in Sec. 1. Then we offer more results on empirical studies (Sec. 2.1), and on different models and modalities (Sec. 2.2, Sec. 2.3, Sec. 2.4), as well as ablation studies including the propagation effect on MLLMs (Sec. 2.5) and the choice of aggregation strategies (Sec. 2.6). Finally we demonstrate how to reproduce our results (Sec. 3).

1. Implementation Details

1.1. FOLDER Architecture for MLLMs

FOLDER is designed as a plug-and-play module plugged in transformer-based vision encoders of Multi-modal Large Language Models (MLLMs), to reduce the output visual token sequence length. FOLDER is a generalized token merging module, that can adapt any matching function or aggregation methods, allowing any number of tokens to be merged in any block without constraint. More specifically, we applied FOLDER between the residual connection of attention block and the MLP. To minimize the calculation overhead for token grouping, we upgrade the bipartite soft matching algorithm to meet the demand for arbitrary number of merging.

Merging Position ToMe [1] or Turbo [7] applies an uniform and progressive token merging, with constant number of reduction on each block to accelerate the vision encoder (ViT [4], CLIP [12], BLIP [8] etc.). Unlike them, we here focus on the acceleration of MLLMs, where the computational cost is concentrated in the LLM. Indeed, for LLaVA1.5-13B [10], the total time to generate the 1st-token is around 0.37s on V100-32G, while the vision encoder part is around 0.03s, which is less than $1/12$. For all the experiments in the main paper, we only reduce tokens in the output layer/block of vision encoder (LLaVA1.5 [10] uses the image feature of the second last layer, while Minigpt4v2 [2] uses only the last layer). The ablation on reduction partition between blocks conducted on MLLMs is provided in Tab. 4. Similar to the result on BLIP in the main paper, reducing tokens only in the last layer yields the best result, which is in line with our empirical observation.

Matching Function In FOLDER algorithm, we need to choose a matching function that can evaluate the similarity between tokens, so that we aggregate tokens with similar semantic meanings. ToMe [1] directly calculates the cosine similarity between tokens' key value in attention calculation (K taking the mean on multi-head), while Turbo [7] leverage a more delicate matching function that considers both similarity between tokens and the semantic importance of tokens (attention contribution for the class token). For the experiment in the main paper, we adopt the matching function of Turbo, by replacing the metric of key value by token itself. To minimize the implementation effort, we offer an extremely simplified version that only evaluate the cosine similarity between tokens in the last layer (between attention and MLP), and the performance gap is minor (please refer to the ablation studies in section 2.6). This allows the adaptation to be a ready-to-use on any MLLM visual backbones.

Merging Order To realize the average merging that is independent of the folding order, we make one little adjustment. For example, if we have two folding operations, which asks for token (x_1, x_3, x_5) to be merged as x_8 in the first fold, and (x_6, x_7, x_8) to be merged in the second fold. We would like to average on $(x_1, x_3, x_5, x_6, x_7)$, without taking the merging order into account.

$$x_{\text{merge}} = \text{avg}(x_1, x_3, x_5, x_6, x_7)$$

To do this, we use a size list to note the number of tokens that contributed to obtaining the merged token (for the merged token x_8 after the first fold, the corresponding size is 3), then we weight the token by their size during the fol-

Method	Reduct Ratio	Speed Up	MMBench EN	MM Star	MME	A-OK VQA	Hallusion Bench	MM MU	POPE	SEED BenchIMG	Science QA	RealW QA	Avg
Original	0%	1×	9.2	24.4	631.4	38.5	26.4	18.7	62.3	31.8	48.6	38.8	33.0
Turbo	50%	1.3×	8.7	21.0	692.1	35.0	**34.7**	20.7	52.7	30.5	50.4	37.6	32.6
Ours		1.3×	**9.4**	**22.5**	**710.3**	**38.4**	34.6	**24.0**	**56.2**	**31.3**	**50.7**	**38.6**	**34.1**
Turbo	60%	1.4×	9.1	22.1	674.5	36.7	34.3	23.0	51.0	30.4	50.3	38.8	33.0
Ours		1.4×	**13.8**	**24.3**	**859.9**	**43.8**	**35.6**	**23.1**	**63.3**	**33.7**	**53.5**	**39.9**	**37.4**
Turbo	70%	1.6×	8.7	22.0	651.8	36.3	34.4	20.7	44.7	30.7	50.1	35.0	31.5
Ours		1.6×	**9.3**	**22.5**	**666.8**	**37.6**	**35.5**	**22.8**	**56.6**	**31.2**	**51.5**	**37.9**	**33.9**

Table 1. **Results on Minigpt4v2.** We highlight the best result on each benchmark under the same reduction ratio.

Method	MMB	A-OKVQA	MME	MMMU	SEEDBench	SQA	AI2D
Idefics2-8B	76.4	83.6	1497.8	42.4	72.2	86.8	72.5
+FOLDER-70%	76.0	83.1	1491.3	42.8	72.0	86.9	72.0
Flamingov2-7B	7.0	28.0	530.8	26.8	28.7	43.7	31.3
+FOLDER-70%	7.0	28.6	542.0	26.6	28.5	44.1	31.6

Table 2. **FOLDER on non-LLaVA models (All above are training-free).**

Method	M40.Classfication(I)	Obj3000.Classfication(I)	Obj3000.Captioning(GPT-4)
PointLLMv1.2-7B	53.5	45.7	42.5
+FOLDER-70%	51.3	45.0	41.7

Table 3. **FOLDER on PointLLM with 70% reduction ratio & 1.4× speed-up.**

lowing fold (for token x_8, we considered $3 \times x_8$ during the average computation with (x_6, x_7):

$$x_8 = \text{avg}(x_1, x_3, x_5)$$

$$x_{\text{merge}} = (3 \times x_8 + x_6 + x_7)/(3 + 1 + 1)$$

In this way, we realize average merging regardless of the folding order. We release the code on BLIP and LLaVA. For more implementation details, please refer to the code along with this file.

Order Rearrange and Size Compensation Since all the visual tokens will be assigned new positional encoding inside LLM, we want to preserve the relative position between tokens. In the algorithm presented in the main paper, we partition $\mathbb{A}$ and $\mathbb{B}$ crossly, so when concatenating $\mathbb{A}'$ and $\mathbb{B}'$, we recover the original order of each token. Besides, several tokens are merged as one during this process, which may degrade the importance score in LLM attention. To mitigate this behavior, we save the size of each merged token and multiply them by $1 + log(size)$ when outputting from vision backbone. We adopt size compensation and order rearrange for simplified version of FOLDER. More details can be found in the code repository.

1.2. Training Details of LLaVA1.5-13B

To test the effectiveness of FOLDER for training phase, we train LLaVA1.5-13B following the same procedures (pretrain-sft two-stage training, dataset, training parameters) detailed in LLaVA1.5 repository, with FOLDER inserted in the visual backbone of LLaVA1.5. The whole training process is conducted on 8 A100-80G GPUs and the GPU hour in Tab. 6 in the main paper is evaluated by the actual training time multiplying the number of GPUs.

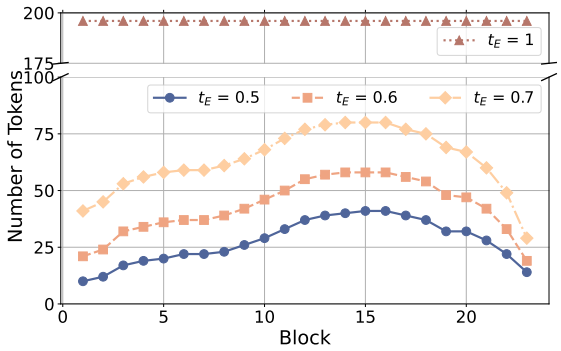

Figure 1. **Minimum Number of Tokens with Energy t_E Across Blocks on ViT Large.** We evaluate on 3 different t_E for every block on ViT large 24 blocks.

1.3. Reproduction Details of Baselines

We reproduce several SOTA MLLM token reduction methods [3, 7, 13–16]. For FastV, we drop visual tokens on the 3rd layer of LLM. For sparsevlm [16], we set scale=13.5 and bias=0. For pyramiddrop [15], we set the layer list to be [4,12,20] and reduction ratio list to be [0.3,0.2,0.1] for both training and inference. For LLaVA-Prumerge [13], we use static 1/4 reduction for fair comparison. For Crossget [14] and Turbo [7], we adopt uniform reduction across all layers, and calculate the reduction ratio based on the remaining tokens at the output. LLaVA-Prumergeand Crossget results are reproduced using official checkpoints while pyramiddrop and pixel-shuffle are trained by us (the others are training-free methods).

2. More Experiments

2.1. Results on Empirical Studies

In addition to the empirical results on ViT-B, we also conduct such experiments on ViT-S and ViT-L to demonstrate the generality of such phenomenon. As shown in Fig. 1, 2, 3 and 4, on models of various sizes, the trend of Energy & EMD distance with respect to blocks is similar. Combined with the results in Tab. 4, we can conclude that merging on last layers is the best choice.

Method	Reduct Ratio	MMBench EN	MM Star	MME	A-OK VQA	Hallusion Bench	MM MU	OCR VQA	SEED BenchIMG	Science QA	POPE	Avg
Original-7B	0%	62.8	32.7	1338.9	78.8	35.6	32.2	52.4	60.2	68.1	79.7	55.0
Uniform	66%	60.1	**31.5**	1301.6	**78.0**	34.9	24.7	34.4	57.8	67.2	85.0	52.0
Last-3		60.9	31.1	1312.4	77.4	36.4	**31.6**	41.9	58.5	67.9	85.7	53.9
Last-2		61.1	30.4	**1353.1**	76.9	35.9	31.4	41.6	58.4	67.9	**85.7**	53.8
Last-1		**61.4**	30.3	1350.0	77.9	**39.2**	31.3	**46.1**	**59.7**	**68.3**	85.4	**54.8**
Original-13B	0%	66.6	30.9	1371.1	77.0	36.1	34.0	54.9	59.4	68.8	86.4	56.3
Uniform	66%	64.7	31.2	1168.3	76.3	25.9	33.3	47.9	58.5	70.5	85.5	53.6
Last-3		**66.5**	31.6	**1369.4**	77.2	34.5	34.8	50.1	58.7	**70.9**	**86.5**	55.9
Last-2		65.7	**32.5**	1332.0	77.2	**34.6**	34.5	51.7	58.6	70.3	86.0	55.9
Last-1		65.8	31.7	1366.9	**77.3**	33.8	**35.0**	**52.6**	**58.8**	70.7	86.1	**56.1**

Table 4. **Propagation Effect on LLaVA1.5 7B/13B.** We evaluate different merging positions using LLaVA1.5 under 66% reduction ratio. Last-n refers to uniform reduction in last n blocks, uniform refers to uniform reduction in every block.

Method	Reduct Ratio	MMBench EN	MM Star	MME	A-OK VQA	Hallusion Bench	MM MU	OCR VQA	SEED BenchIMG	Science QA	POPE	Avg
Original-7B	0%	62.8	32.7	1338.9	78.8	35.6	32.2	52.4	60.2	68.1	79.7	55.0
Direct Drop	66%	39.1	20.7	980.5	64.4	20.1	14.0	11.5	45.6	45.3	77.9	37.4
Weighted Avg		60.5	**30.9**	1332.3	77.7	38.8	31.3	**46.6**	**59.8**	68.2	**85.6**	54.7
Avg		**61.4**	30.3	**1350.0**	**77.9**	**39.2**	**31.3**	**46.1**	59.7	**68.3**	85.4	**54.8**
Original-13B	0%	66.6	30.9	1371.1	77.0	36.1	34.0	54.9	59.4	68.8	86.4	56.3
Direct Drop	66%	43.1	20.6	1082.6	65.5	18.2	19.3	29.9	45.8	47.3	79.5	40.8
Weighted Avg		65.3	31.3	**1374.2**	**77.4**	**35.0**	34.6	52.5	58.3	70.3	85.8	56.0
Avg		**65.8**	**31.7**	1366.9	77.3	33.8	**35.0**	**52.6**	**58.8**	**70.7**	**86.1**	**56.1**

Table 5. **Aggregation method on LLaVA1.5 7B/13B.** We test the performance of 3 aggregation operations. For fair comparison, we adopt the aggregation only in the last block.

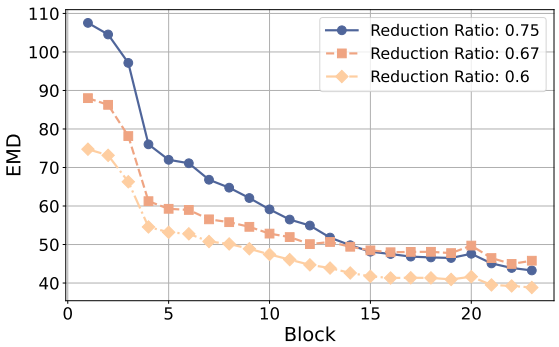

Figure 2. **EMD Distance Between Reduced and Original Output Distributions under 3 Reduction Ratios on ViT Large.** We compare the EMD distance by exerting token reduction on different blocks on ViT large 24 blocks.

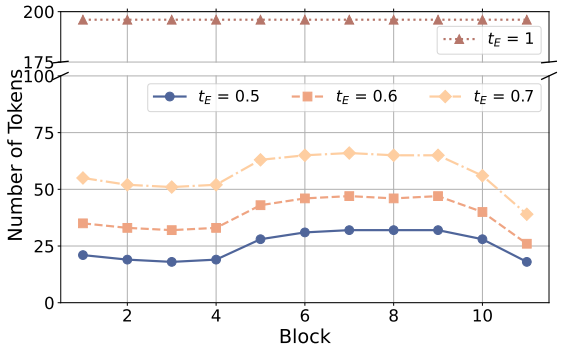

Figure 3. **Minimum Number of Tokens with Energy t_E Across Blocks on ViT Small.** We evaluate on 3 different t_E for every block on ViT small 12 blocks.

2.2. Results on Minigpt-4v2

In Tab. 1, we also evaluate FOLDER on another MLLM: Minigpt4v2. With a reduction ratio up to 60%, our method achieves an overall performance improvement compared to the original model, with an increase of more than 40% on MME and MMBench. Even in this case, by merging in the last block, FOLDER outperforms Turbo. This implies that, in addition to speed enhancement, FOLDER can potentially be a plug-and-play performance booster, reducing the noise occurring in long token sequences.

2.3. Results on Non-LLaVA models

In Tab 2, we offer results on Flamingo and IDEFICS to show our generalizability, which is even better than that on LLaVA-like models.

Method	Reduct Ratio	MMBench EN	MM Star	MME	A-OK VQA	Hallusion Bench	MM MU	OCR VQA	SEED BenchIMG	Science QA	POPE	Avg
Original-7B	0%	62.8	32.7	1338.9	78.8	35.6	32.2	52.4	60.2	68.1	79.7	55.0
$\alpha = 0$	66%	60.9	30.7	1354.1	76.9	39.2	31.5	45.3	59.8	68.0	**85.9**	54.7
$\alpha = 3$		60.8	**30.7**	1341.4	77.7	39.1	31.2	**46.4**	**59.8**	**68.5**	85.7	54.8
$\alpha = 5$		**61.4**	30.3	1350.0	77.9	39.2	31.3	46.1	59.7	68.3	85.4	54.8
Metric $= K$		61.3	30.3	**1354.9**	**78.1**	**39.4**	**32.4**	45.2	59.6	68.3	85.3	**54.8**
Original-13B	0%	66.6	30.9	1371.1	77.0	36.1	34.0	54.9	59.4	68.8	86.4	56.3
$\alpha = 0$	66%	65.3	31.7	1351.4	**77.9**	33.9	**35.2**	52.5	58.5	70.7	86.1	56.0
$\alpha = 3$		65.1	**31.9**	**1383.9**	77.4	**34.7**	34.8	**52.6**	58.4	70.4	86.0	56.1
$\alpha = 5$		**65.8**	31.7	1366.9	77.3	33.8	35.0	52.6	**58.8**	70.7	**86.1**	**56.1**
Metric $= K$		65.4	31.7	1359.5	77.8	33.1	34.0	51.4	58.2	**70.9**	85.4	55.6

Table 6. **Ablation for Matching Functions on LLaVA1.5 7B/13B.** We evaluate various matching functions evolved from ToMe [1] and Turbo [7]. The default setting is $\alpha = 5$ in Eq. 1 and metric as token itself. Metric $= K$ means that we use the key value to calculate cosine similarity between tokens.

Method	Reduct Ratio	MMBench EN	MM Star	MME	A-OK VQA	Hallusion Bench	MM MU	OCR VQA	SEED BenchIMG	Science QA	POPE	Avg
Original-7B	0%	62.8	32.7	1338.9	78.8	35.6	32.2	52.4	60.2	68.1	79.7	55.0
Simplified	66%	60.7	**30.8**	1341.1	76.4	38.8	**31.6**	**46.4**	**60.1**	**68.4**	**86.3**	54.7
Ours-standard		**61.4**	30.3	**1350.0**	**77.9**	**39.2**	31.3	46.1	59.7	68.3	85.4	**54.8**
Original-13B	0%	66.6	30.9	1371.1	77.0	36.1	34.0	54.9	59.4	68.8	86.4	56.3
Simplified	66%	65.4	**31.9**	1357.9	**77.7**	**34.3**	34.7	52.2	58.6	**70.8**	86.1	56.0
Ours-standard		**65.8**	31.7	**1366.9**	77.3	33.8	**35.0**	**52.6**	**58.8**	70.7	**86.1**	**56.1**

Table 7. **Simplified Version of FOLDER on LLaVA1.5 7B/13B.** By directly applying FOLDER on the output of visual encoder, we achieve similar performance with respect to standard FOLDER, while greatly reduce the implementation effort.

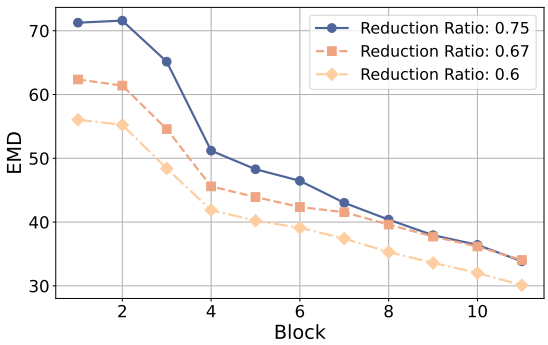

Figure 4. **EMD Distance Between Reduced and Original Output Distributions under 3 Reduction Ratios on ViT Small.** We compare the EMD distance by exerting token reduction on different blocks on ViT small 12 blocks.

2.4. Results on 3D Points Cloud

We also apply FOLDER on PointLLM for 3D point cloud in Tab. 3.

2.5. Ablation Study on Propagation Effect

In Tab. 8 of the main paper, we study the propagation effect on BLIP [8]. In Tab. 4, we offer results on LLaVA1.5-13B, with 4 different reduction partitions (keep the number of remaining tokens unchanged). Reduction in the last layer remains the best strategy for MLLMs of different sizes. Although more subtle partitions can be explored, merging only in the last layer is a simple and safe choice.

2.6. Ablation Study on Aggregation Strategy

Aggregation Method. In Tab. 5, we evaluate the three aggregation methods on MLLMs discussed in the main paper. More specifically, we conduct the experiment on LLaVA1.5-7B and 13B to fully compare these aggregation methods. As shown in Tab. 5, there is a significant reduction in performance using direct drop, especially on dense tasks like OCRVQA. This suggests that direct drop may cause severe information loss. While for merging strategies, weighted average on norm and vanilla average merging both shows superior performance over direct dropping. The performance gap between is minor. For simplicity, we use average merging in as our default aggregation method.

Matching Function. In addition to the aggregation method, we also conduct experiments on matching functions. Turbo [7] proposed a generalized matching function that considers both mutual redundancy (token similarity) and semantic values (attention importance), which is formulated as:

$$\mathcal{E} = \mathcal{R} - \alpha\mathcal{I}, \tag{1}$$

where $\mathcal{R}$ the similarity between tokens and $\mathcal{I}$ token's attention contribution with respect to the class token. α is a weighted hyper-parameter which we take $\alpha = 5$ (a rough

approximation for $\alpha = \text{seq\_len}//100$). To calculate the similarity between tokens, ToMe and Turbo [7] leverage the K (key) in the attention by taking mean on multi-head dimension, thus to save computational cost and enhance slightly the performance. In our experiment, we simply take the token itself as the metric to calculate the cosine similarity for simplicity. In Tab. 6, we ablate on different matching functions. By using various α values and metrics on LLaVA1.5-7B and 13B, the performance rests similar, indicating the robustness of FOLDER.

Simplified Version. Furthermore, in order to avoid all potential difficulties in implementing FOLDER (*e.g.*, FOLDER needs to be inserted into vision backbone, which requires to adapt for different vision encoder's architecture), we introduce one extremely simplified version that only applies FOLDER on the output visual sequence from vision backbone, regardless of the model architecture. In Tab. 7, we show that such a simplified version achieves almost the same performance as standard ones, further showing the stability and universality of our proposed method.

3. Reproducibility

Along with this file, we offer the implementation of complete FOLDER on BLIP and LLaVA1.5, equipped with VLMEvalKit [5] to evaluate on various MLLM benchmarks. We also offer examples of simplified FOLDER inserted in 3 SOTA MLLMs (VILA1.5 [9], WePOINTS1.5 [11], VITA1.5 [6]). For more details, please refer to the provided code.

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. *arXiv preprint arXiv:2210.09461*, 2022. 1, 4

[2] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478*, 2023. 1

[3] Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. *arXiv preprint arXiv:2403.06764*, 2024. 2

[4] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[5] Haodong Duan, Junming Yang, Yuxuan Qiao, Xinyu Fang, Lin Chen, Yuan Liu, Xiaoyi Dong, Yuhang Zang, Pan Zhang, Jiaqi Wang, et al. Vlmevalkit: An open-source toolkit for evaluating large multi-modality models. In *Proceedings of the 32nd ACM international conference on multimedia*, pages 11198–11201, 2024. 5

[6] Chaoyou Fu, Haojia Lin, Xiong Wang, Yi-Fan Zhang, Yunhang Shen, Xiaoyu Liu, Yangze Li, Zuwei Long, Heting Gao, Ke Li, et al. Vita-1.5: Towards gpt-4o level real-time vision and speech interaction. *arXiv preprint arXiv:2501.01957*, 2025. 5

[7] Chen Ju, Haicheng Wang, Haozhe Cheng, Xu Chen, Zhonghua Zhai, Weilin Huang, Jinsong Lan, Shuai Xiao, and Bo Zheng. Turbo: Informativity-driven acceleration plug-in for vision-language large models. In *European Conference on Computer Vision*, pages 436–455. Springer, 2025. 1, 2, 4, 5

[8] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 1, 4

[9] 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. 5

[10] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26296–26306, 2024. 1

[11] Yuan Liu, Le Tian, Xiao Zhou, Xinyu Gao, Kavio Yu, Yang Yu, and Jie Zhou. Points1. 5: Building a vision-language model towards real world applications. *arXiv preprint arXiv:2412.08443*, 2024. 5

[12] 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. 1

[13] 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. 2

[14] Dachuan Shi, Chaofan Tao, Anyi Rao, Zhendong Yang, Chun Yuan, and Jiaqi Wang. Crossget: Cross-guided ensemble of tokens for accelerating vision-language transformers. *arXiv preprint arXiv:2305.17455*, 2023. 2

[15] Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, et al. Pyramiddrop: Accelerating your large vision-language models via pyramid visual redundancy reduction. *arXiv preprint arXiv:2410.17247*, 2024. 2

[16] Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, et al. Sparsevlm: Visual token sparsification for efficient vision-language model inference. *arXiv preprint arXiv:2410.04417*, 2024. 2