# Growing a Twig to Accelerate Large Vision-Language Models

Supplementary Material

Zhenwei Shao[1*] Mingyang Wang[1*] Zhou Yu[1†] Wenwen Pan[1] Yan Yang[1]
Tao Wei[2] Hongyuan Zhang[2] Ning Mao[2] Wei Chen[2] Jun Yu[1, 3]

[1] Zhejiang Key Laboratory of Space Information Sensing and Transmission,
School of Computer Science, Hangzhou Dianzi University, China
[2]Li Auto Inc., China [3]School of Intelligence Science and Engineering, Harbin Institute of Technology (Shenzhen), China

## A. More Implementation Details

**TwigVLM training.** As described in the main text, the twig block is trained by finetuning the shallow VLM $\mathcal{M}_s$. Specifically, $\mathcal{M}_s$ is initialized with the weights of the first $K$+$T$ layers and the prediction head of the corresponding base VLM $\mathcal{M}_b$. During finetuning, only the last $T$ layers and the prediction head—collectively termed the twig block—are updated, while the first $K$ layers remain frozen. This process follows the same training manner to train the base VLM $\mathcal{M}_b$. Theoretically, any suitable multimodal instruction tuning dataset can be employed to finetune $\mathcal{M}_s$.

In our experiments, we use the LLaVA-665K dataset [13] to train the twig blocks for the LLaVA-1.5-7B and LLaVA-NeXT-7B models, and combine the LLaVA-665K and VideoInstruct-100K datasets [11, 17] for the Video-LLaVA-7B model. The optimization hyper-parameters are detailed in Table 1. All training is performed on a server equipped with 8 NVIDIA A100 GPUs. Under these conditions, the training is highly efficient, requiring only approximately 10% of the time needed to train the corresponding base VLM, *e.g.*, training the twig block for the LLaVA-1.5-7B model takes about 10 GPU hours, while the training of the original LLaVA-1.5-7B takes about 100 GPU hours.

**Twig-guided token pruning (TTP).** During inference, TwigVLM leverages the TTP strategy to perform token pruning over the base VLM: (i) at the $K$-th layer, selecting $R$ key visual tokens (output by the $K$-th layer) and discarding the rest tokens guided by the attention map from the last twig layer, and (ii) applying the FinalWipe strategy to further remove all the visual tokens after the $K_\text{f}$-th layer. Therefore, we adjust the value of $R$ to satisfy different pruning ratios calculated by the average number of retained visual tokens $\bar{R}$. Table 2 shows the default pruning settings for TwigVLM under different pruning ratios.

| config | setting |
|---|---|
| optimizer | AdamW |
| weight decay | 0. |
| optimizer momentum | $\beta_1, \beta_2$=$0.9, 0.98$ |
| batch size | 128 |
| learning rate schedule | cosine decay |
| peak learning rate | 5e-5 |
| warm-up strategy | linearly warm-up |
| warm-up ratio | 0.03 |
| training epochs | 1 |

Table 1. **Training settings.** These hyper-parameters are shared across all the TwigVLM models in our experiments.

| $\bar{R}$ | pruning ratio | $K$ | $R$ | $K_\text{f}$ |
|---|---|---|---|---|
| 192 | 66.7% | 2 | 227 | 24 |
| 128 | 77.8% | 2 | 134 | 24 |
| 64 | 88.9% | 2 | 41 | 24 |

Table 2. **Pruning settings.** These hyper-parameters correspond to the default TTP settings of different pruning ratios.

**Self-speculative decoding (SSD).** For efficient generation of long responses, TwigVLM applies the SSD strategy by using the $\mathcal{M}_s$ as the *draft* model and $\mathcal{M}_b$ as the *target* model. Specifically, in each SSD iteration, the draft model efficiently predicts $\delta = 5$ subsequent draft tokens in an autoregressive manner. To further improve efficiency, this draft generation process is equipped with an early-exit mechanism that allows the draft model to stop generation if the probability of the current predicted token falls below a predefined threshold $\theta = 0.6$. The target model then verifies these generated draft tokens in parallel, accepts those matching the target model's predictions, and then predicts a next token by itself. The iteration repeats until the <EOS> token is generated. Note that the TTP and SSD strategies

[*] Work done during an internship at Li Auto Inc.

[†] Zhou Yu is the corresponding author

**Algorithm 1** Pseudocode of TwigVLM's inference process

```
# bVLM: the base VLM model, i.e., M_b
# twig: the twig block
# K: Number of shared low layers
# K_f: The position to apply FinalWipe
# R: Number of retained visual tokens when pruning
# delta: Maximum draft token length
# theta: Confidence threshold to stop draft

def sVLM_forward(tokens):
    X_k = bVLM.forward_low_layers(tokens, k=K)
    prob, Attn_last = twig.forward(X_k)
    a_i = argmax(prob)
    return X_k, prob, Attn_last, a_i


def TwigVLM_inference(img, ques):
    draft_toks = [] # temporary buffer for draft tokens
    final_resp = [] # buffer for final response

    # Prefilling stage of sVLM
    X_k, _, Attn, a_i = sVLM_forward((img, ques))
    draft_toks.append(a_i)

    # Prune visual tokens in X_k using Eq. (5)
    # X_k_b means shared token latents for bVLM
    X_k_b = pruning(X_k, Attn, r=R)

    # The loop of self speculative decoding
    while EOS_TOKEN not in final_resp:
        X_k, prob, _, a_i = sVLM_forward(a_i)
        draft_toks.append(a_i)
        X_k_b = concat(X_k_b, X_k, axis=1)

        # the condition to stop draft and verify
        if len(draft_toks) >= delta or prob < theta:
            # removing all visual tokens after layer K_f
            tgt_probs = bVLM.forward_high_layers(
                X_k_b, k=K, fianl_wipe=K_f)
            # verification
            right_toks = [a for a, p in zip(draft_toks, tgt_probs
                [:-1]) if argmax(p) == a]
            right_toks.append(argmax(tgt_probs[-1]))
            final_resp.extend(right_tokens)
            # reset temporary variables
            draft_toks = []
            X_k_b = None
            a_i = final_resp[-1]
    return final_resp
```

can be seamlessly integrated, as detailed in Algorithm 1.

## B. More Experimental Results

### B.1. More performance comparisons

**Comparisons on more benchmarks.** Taking LLaVA-1.5-7B as the base VLM, Table 4 compares the accuracies between TwigVLM and other visual token pruning methods on *nine* VLM benchmarks under three different pruning ratios. TwigVLM consistently outperforms or matches its counterparts on all benchmarks and pruning ratios, achieving the best overall RelAcc. In particular, TwigVLM even surpasses the upper bound given by the base VLM in RelAcc (100.6%) with a 66.7% pruning ratio, demonstrating its spectacular effectiveness and robustness in accelerating VLMs to deal with various tasks.

**Comparisons on on more base VLMs.** To further demonstrate the generalization ability and superiority of TwigVLM, we present additional experimental results on a larger VLM, LLaVA-1.5-13B, and a more recent VLM, Qwen2.5-VL-7B, which is very capable and representative.

| Method | GQA | MMB | MME | VQA$^{T}$ | SQA$^{I}$ | VQA$^{V2}$ | RelAcc |
|---|---|---|---|---|---|---|---|
| *Upper Bound, 576 Tokens* **(100%)** | | | | | | | |
| LLaVA-1.5-13B | 63.2 | 67.7 | 1818 | 61.3 | 72.8 | 80.0 | 100% |
| *Retain Averaged 192 Tokens* (↓ **66.7%**) | | | | | | | |
| FastV | 60.3 | 67.4 | 1807 | 60.4 | **74.0** | 77.7 | 98.6% |
| VisionZip | 59.1 | 66.9 | 1754 | 59.5 | 73.5 | 78.1 | 97.4% |
| VisionZip‡ | 61.6 | 67.1 | 1790 | 59.9 | 72.7 | 78.6 | 98.5% |
| **TwigVLM (ours)** | **62.5** | **68.6** | **1840** | **60.4** | 73.1 | **79.4** | **99.9%** |
| *Retain Averaged 128 Tokens* (↓ **77.8%**) | | | | | | | |
| FastV | 57.5 | 65.9 | 1758 | 58 | 73.8 | 74.3 | 95.7% |
| VisionZip | 57.9 | 66.7 | 1743 | 58.7 | **74.0** | 76.8 | 96.6% |
| VisionZip‡ | 60.1 | **67.6** | 1736 | 59.2 | 73.0 | 77.6 | 97.4% |
| **TwigVLM (ours)** | **61.2** | 66.9 | **1811** | **60.2** | 73.4 | **79.1** | **98.9%** |
| *Retain Averaged 64 Tokens* (↓ **88.9%**) | | | | | | | |
| FastV | 50.1 | 55.9 | 1408 | 52.2 | 73.2 | 61.1 | 83.6% |
| VisionZip | 56.2 | 64.9 | 1676 | 57.4 | **74.4** | 73.7 | 94.2% |
| VisionZip‡ | 58.1 | 65.6 | 1671 | **58.5** | 72.3 | 75.2 | 94.9% |
| **TwigVLM (ours)** | **60.0** | **67.4** | **1765** | 58.4 | 72.4 | **77.0** | **97.1%** |

Table 3. Performance comparisons of TwigVLM with other token pruning methods on **LLaVA-1.5-13B**.

Table 3 provides the accuracy comparisons on the larger LLaVA-1.5-13B model. TwigVLM consistently achieves the best overall RelAcc compared to all the counterparts, with its superiority being more significant as the increase of pruning ratios. These results verify the scalability and generalization ability of TwigVLM in accelerating large VLMs.

To evaluate the universality of TwigVLM to VLMs beyond the LLaVA family, we also apply it to Qwen2.5-VL-7B. Since the SFT data for QwenVL is large-scale yet inaccessible, we use a 5M subset of the MAmmoTH-VL-10M dataset (single image split) [6] as an alternative to train TwigVLM, which takes 12 hours on 8*A100 NVIDIA GPUs. The results in Table 5 show that TwigVLM can still maintain its prominent advantages over FastV in terms of accuracy and speed, showing the effectiveness of TwigVLM in practical scenarios.

### B.2. More ablation studies

**Token acceptance rate in SSD.** In the context of speculative decoding methods [3, 9, 12], the token acceptance rate (*abbr.* TokAR) serves as a critical metric for assessing the efficacy of these approaches. TokAR is defined as the proportion of the draft tokens generated by the draft model that are subsequently accepted by the target model. In TwigVLM, TokAR plays a key role, which is influenced by the effectiveness of the twig block and has a significant impact on model's generation speed.

To analyze how TokAR is influenced by the design choices in TwigVLM, we evaluate this metric on several representative variants from the ablation studies presented

| Method | GQA | MMB | MME | VQA$^{T}$ | SQA$^{I}$ | VQA$^{V2}$ | POPE | MMMU | MM-Vet | RelAcc |
|---|---|---|---|---|---|---|---|---|---|---|
| *Upper Bound, 576 Tokens* **(100%)** | | | | | | | | | | |
| LLaVA-1.5-7B | 61.9 | 64.7 | 1862 | 58.2 | 69.5 | 78.5 | 85.9 | 36.3 | 31.1 | 100% |
| *Retain Averaged 192 Tokens* (↓ **66.7**%) | | | | | | | | | | |
| FastV [2] | 56.5 | 63.7 | 1786 | 57.3 | 69.5 | 74.6 | 79.2 | 35.7 | 28.1 | 95.6% |
| SparseVLM [26] | 57.6 | 62.5 | 1721 | 56.1 | 69.1 | 75.6 | 83.6 | 33.8 | 31.5 | 96.2% |
| PDrop [19] | 57.3 | 63.3 | 1797 | 56.5 | 69.2 | 75.1 | 82.3 | - | - | 96.4% |
| MustDrop [14] | 58.2 | 62.3 | 1787 | 56.5 | 69.2 | 76.0 | 82.6 | - | - | 96.6% |
| VisionZip [21] | 59.3 | 63.0 | 1783 | 57.3 | **68.9** | 76.8 | 85.3 | **36.6** | 31.7 | 98.5% |
| VisionZip‡ [21] | 60.1 | 63.4 | 1834 | 57.8 | 68.2 | 77.4 | 84.9 | 36.2 | 32.6 | 99.2% |
| **TwigVLM (ours)** | **61.2** | **64.0** | **1848** | **58.0** | 68.8 | **78.1** | **87.2** | **36.6** | **34.1** | **100.8%** |
| *Retain Averaged 128 Tokens* (↓ **77.8**%) | | | | | | | | | | |
| FastV | 53.0 | 61.4 | 1646 | 56.0 | 69.5 | 69.2 | 73.2 | 36.3 | 28.0 | 92.1% |
| SparseVLM | 56.0 | 60.0 | 1696 | 54.9 | 67.1 | 73.8 | 80.5 | 33.8 | 30.0 | 93.6% |
| PDrop | 57.1 | 61.6 | 1761 | 56.6 | 68.4 | 72.9 | 82.3 | - | - | 95.2% |
| MustDrop | 56.9 | 61.1 | 1745 | 56.3 | 68.5 | 74.6 | 78.7 | - | - | 94.6% |
| VisionZip | 57.6 | 62.0 | 1762 | 56.8 | 68.9 | 75.6 | 83.2 | **37.9** | 32.6 | 98.1% |
| VisionZip‡ | 58.9 | 62.6 | **1823** | 57.0 | 68.3 | 76.6 | 83.7 | 37.3 | **32.9** | 98.8% |
| **TwigVLM (ours)** | **60.6** | **63.5** | 1818 | **57.8** | **69.5** | **77.9** | **86.6** | 36.6 | 32.8 | **99.9%** |
| *Retain Averaged 64 Tokens* (↓ **88.9**%) | | | | | | | | | | |
| FastV | 44.1 | 45.9 | 1218 | 50.7 | 70.0 | 52.0 | 55.6 | 34.0 | 17.8 | 75.3% |
| SparseVLM | 52.7 | 56.2 | 1505 | 51.8 | 62.2 | 68.2 | 75.1 | 32.7 | 23.3 | 85.6% |
| PDrop | 47.5 | 58.8 | 1561 | 50.6 | 69.0 | 69.2 | 55.9 | - | - | 84.4% |
| FasterVLM [25] | 51.5 | 58.5 | 1573 | 53.1 | 69.6 | 66.8 | 67.2 | - | 27.5 | 87.6% |
| MustDrop | 53.1 | 60.0 | 1612 | 54.2 | 63.4 | 69.3 | 68.0 | - | - | 88.1% |
| VisionZip | 55.1 | 60.1 | 1690 | 55.5 | 69.0 | 72.4 | 77.0 | **36.2** | **31.7** | 94.5% |
| VisionZip‡ | 57.0 | **61.5** | 1756 | **56.0** | 68.8 | 74.2 | 80.9 | 35.6 | 30.2 | 95.6% |
| **TwigVLM (ours)** | **58.8** | 60.4 | **1760** | 55.8 | **70.0** | **75.6** | **82.7** | 35.9 | 31.1 | **96.8%** |

Table 4. **Performance of TwigVLM on LLaVA-1.5-7B compared to existing methods** under three different pruning ratios. The best result for each benchmark and pruning ratio is bolded.

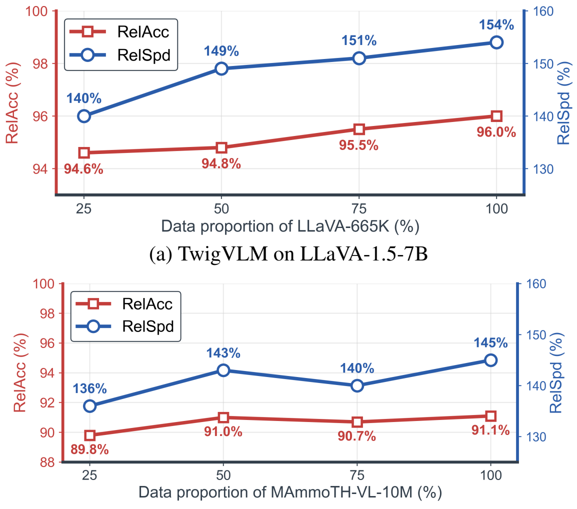
| Method | GQA | MME | MMB | SQA$^{I}$ | VQA$^{T}$ | VQA$^{V2}$ | RelAcc | RelSpd |
|---|---|---|---|---|---|---|---|---|
| *Upper Bound, 1,280 Tokens* **(100%)** | | | | | | | | |
| Q2.5VL-7B | 60.7 | 2347 | 82.7 | 75.3 | 83.2 | 77.9 | 100.0% | 100.0% |
| *Retain Averaged 426 Tokens* (↓ **66.7**%) | | | | | | | | |
| FastV | 57.2 | 2299 | **80.9** | 75.5 | 81.5 | 74.2 | 97.3% | 101.7% |
| **TwigVLM** | **60.4** | **2338** | 79.4 | **77.9** | **82.6** | **78.4** | **99.8%** | **147.7%** |
| *Retain Averaged 284 Tokens* (↓ **77.8**%) | | | | | | | | |
| FastV | 53.5 | **2246** | **78.6** | 75.3 | 79.2 | 70.6 | 94.1% | 103.1% |
| **TwigVLM** | **59.9** | 2238 | 77.4 | **78.1** | **81.4** | **78.4** | **98.3%** | **151.2%** |
| *Retain Averaged 142 Tokens* (↓ **88.9**%) | | | | | | | | |
| FastV | 45.1 | 1859 | 61.5 | 72.8 | 62.9 | 58.2 | 79.1% | 104.3% |
| **TwigVLM** | **57.6** | **2020** | **67.0** | **74.9** | **73.0** | **75.6** | **91.0%** | **152.3%** |

Table 5. Performance comparisons on **Qwen2.5-VL-7B**. The RelAcc and RelSpd are evaluated on the same benchmarks mentioned as the main text.

| ablation variant | **TokAR** (%) | **RelSpd** (%) |
|---|---|---|
| *Twig block initialization* (Table 4c in main text) | | |
| (a) random init. | 37.7 | 120.4 |
| (b) VLM layers[$L$-$T$:$L$] | 44.1 | 131.4 |
| (c) VLM layers[$K$:$K$+$T$] | **57.4** | **153.6** |
| *Number of twig layers* (Table 4d in main text) | | |
| (d) $T$ = 1 | 48.7 | **154.1** |
| (e) $T$ = 2 | 53.4 | 152.6 |
| (f) $T$ = 3 | 57.4 | 153.6 |
| (g) $T$ = 4 | **58.1** | 145.4 |

Table 6. **Token acceptance rate in SSD**. We evaluate the token acceptance rate (TokAR) of the variants in the ablation experiments of the main text.

in the main text. From the results shown in Table 6, we have the following findings: (i) A more effective draft model can be trained by only modifying the initialization strategy without altering the architecture. The variant (c) achieves the highest TokAR (57.4%) and thus the highest generation



(a) TwigVLM on LLaVA-1.5-7B

(b) TwigVLM on Qwen2.5-VL-7B

Figure 1. Performance comparisons of TwigVLM models trained with different proportions of the training dataset. Specifically, we use LLaVA-665K to train TwigVLM models for LLaVA-1.5-7B and use MAmmoTH-VL-10M to train TwigVLM models for Qwen2.5-VL-7B. Even with only 50% of respective training data, TwigVLM is able to maintain competitive accuracy and speed.

speedup. (ii) Increasing the number of twig layers $T$ introduces more computational costs while improving TokAR at the same time. As a result, the RelSpd exhibits only a modest decline when $T$ increases from 1 to 3. However, it drops distinctly at $T$=4, which indicates that TokAR begins to saturate. These findings suggest that TwigVLM achieves higher speedup by striking an optimal balance between TokAR and computation costs of the draft model.

**Data efficiency.** To demonstrate the data efficiency of TwigVLM, we train multiple models using different proportions (i.e., 25%, 50%, 75%, and 100%) of each model's respective training dataset for LLaVA-1.5-7B and Qwen2.5-VL-7B. As shown in Figure 1, both models exhibit a general upward trend in accuracy and speed as the amount of training data increases. Remarkably, however, even when trained on only 50% of their respective datasets, TwigVLM models already achieve competitive, and in some cases comparable, performance to models trained on the full dataset. Moreover, TwigVLM requires only 10% of the training cost of the corresponding base VLM (see A). To sum up, it is highly efficient and feasible to apply TwigVLM in industrial scenarios.

**Memory footprint analysis.** We measure the inference VRAM usage of the LLaVA-1.5-7B and LLaVA-Next-7B models in Table 7. The introduction of the twig block brings 8% extra VRAM cost for loading model weights. Compared to the base VLM, the overall inference VRAM cost of TwigVLM is comparable or slightly reduced due to the substantial reduction of visual tokens.

| model | avg. visual tokens ($\bar{R}$) | model weights VRAM (GB) | inference VRAM (GB) |
|---|---|---|---|
| LLaVA-1.5-7B | 576 | 14.3 | 15.8 |
| + TwigVLM | 64 | 15.5 | 16.5 |
| LLaVA-Next-7B | 2,880 | 14.3 | 17.9 |
| + TwigVLM | 320 | 15.5 | 16.8 |

Table 7. Memory footprint comparisons during inference.

## C. More Visualized Results

In this section, we provide more visualized results to validate the effectiveness of TwigVLM's two key components: the twig-guided visual token pruning (TTP) and self-speculative decoding (SSD). We use LLaVA-1.5-7B as the base VLM in the following experiments.

**Visual token pruning.** To better understand the effectiveness of the proposed TTP strategy, we compare TwigVLM with two representative token pruning methods, namely FastV [2] and VisionZip [21], by visualizing their attention map for token selection and providing the corresponding answer predictions. We provide 16 examples from the GQA and TextVQA benchmarks. As illustrated in Figure 2, TwigVLM demonstrates superior ability to comprehend the semantics in both the textual prompt and image, and accurately identify task-specific image patches (*i.e.*, visual tokens), thereby activating more informative visual tokens for token pruning. In contrast, FastV and VisionZip often fail to capture the fine-grained visual details, leading to suboptimal token selection and incorrect predictions. Notably, even though TwigVLM predicts an incorrect answer, its activated visual tokens according to the attention map is reasonable. This suggests that TwigVLM's occasional failures may not be caused by the visual token pruning, but due to the limitations of the base VLM. These findings verify and explain the effectiveness of the TTP strategy.

**Self-speculative decoding.** To better understand the decoding behavior of the SSD strategy in TwigVLM, we show 8 examples of generated long responses on MM-Vet. From the results in Figure 3, we have two key observations: (i) In general, the proportion of accepted tokens (in green) surpasses that of the corrected tokens (in black) by the target model, indicating that TwigVLM achieves significant speedup through its high token acceptance rate. (ii) The majority of *easy* tokens, such as those associated with grammar and punctuation, are readily accepted. In contrast, the *hard* tokens, which often demand complex reasoning, have a high probability to be corrected by the target model. In practice, the proportion of easy tokens is usually larger than the hard ones, which confirms the the effectiveness of our SSD strategy in accelerating the decoding stage while maintaining the generation quality.

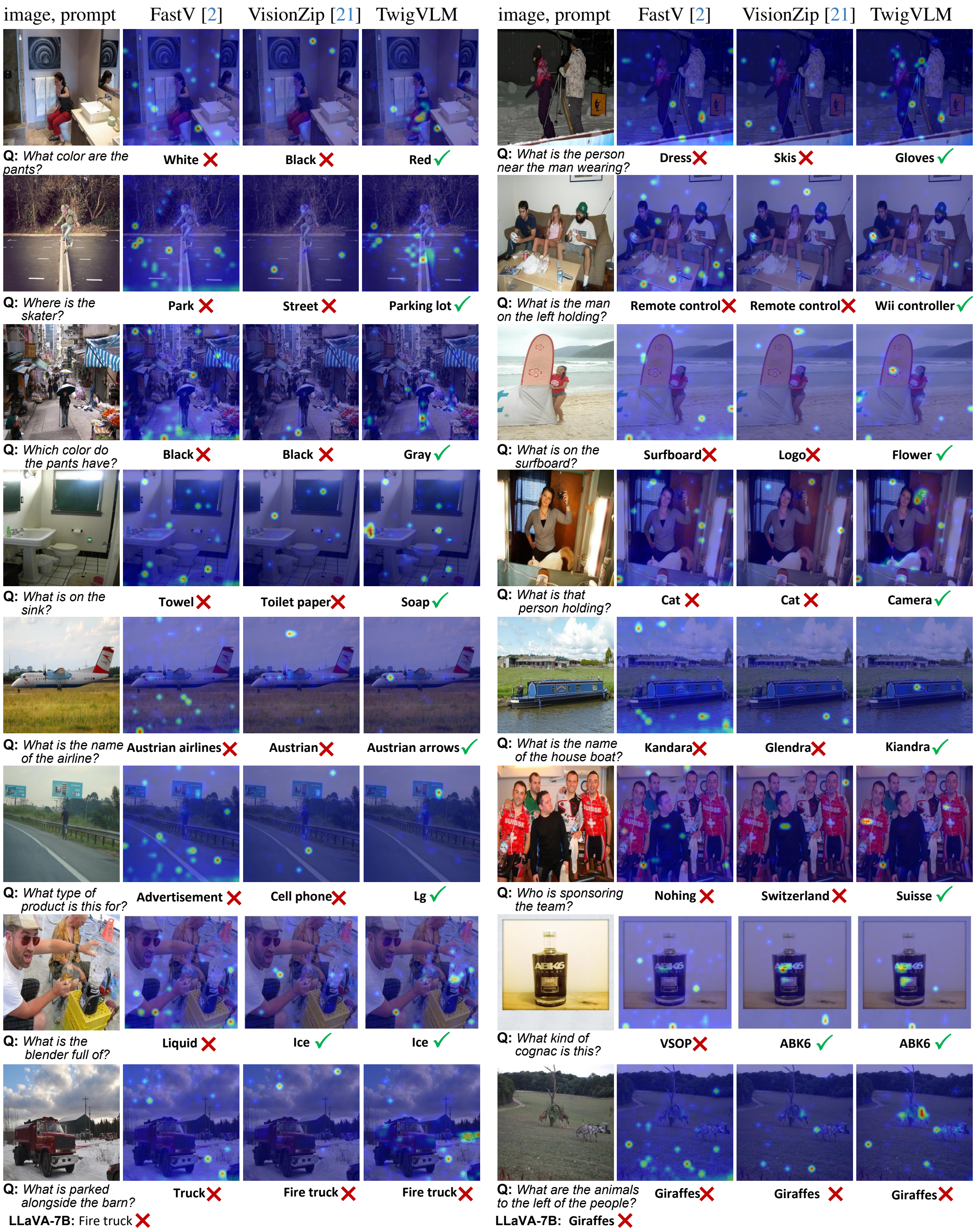
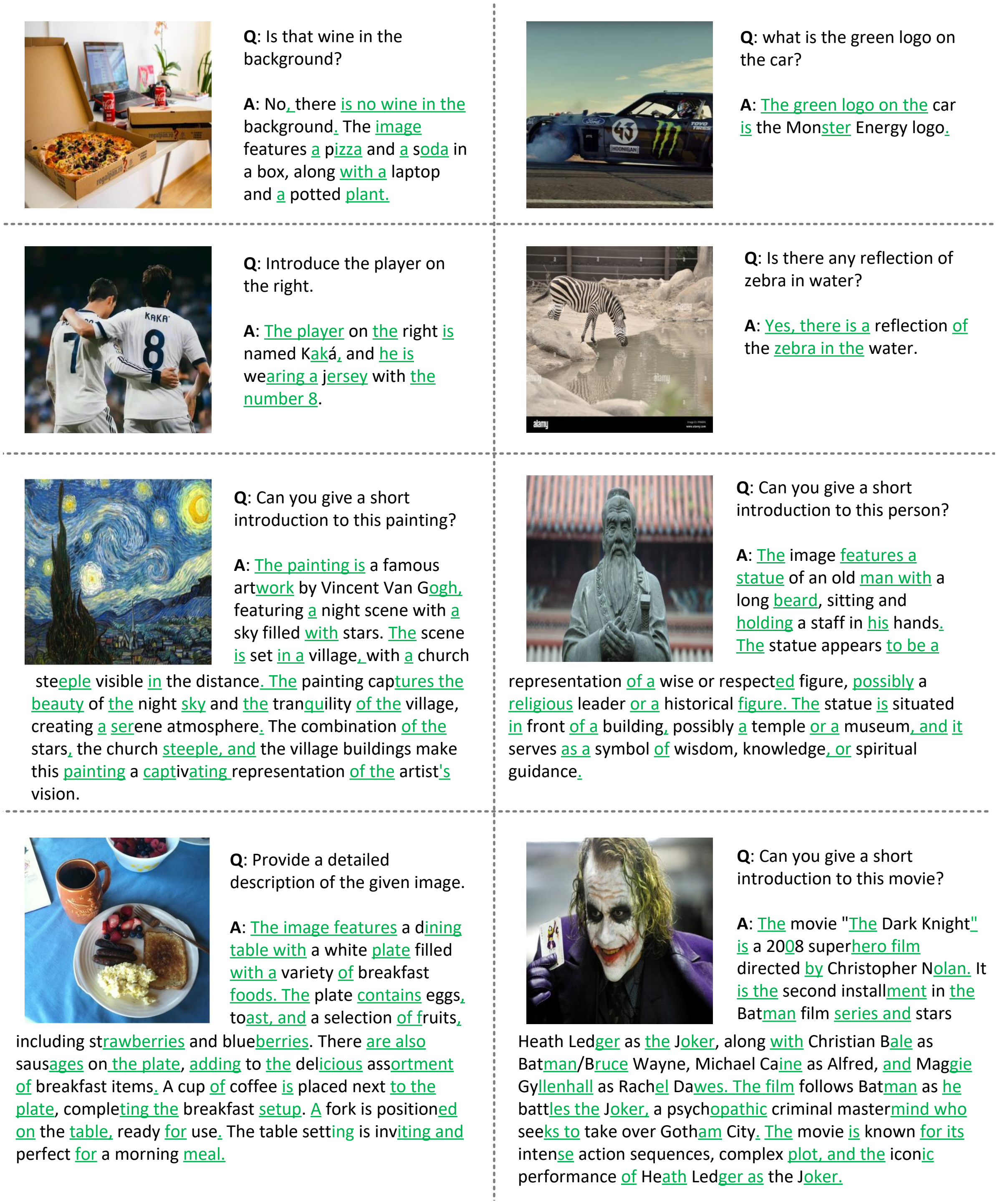


Figure 2. Visualization of attention maps and predictions for FastV [2], VisionZip [21], and our TwigVLM on the examples chosen from the GQA [7] and TextVQA [18]. For the examples in the last row, we additionally provide the predictions from the LLaVA-1.5-7B.



**Q**: Is that wine in the background?

**A**: No, there is no wine in the background. The image features a pizza and a soda in a box, along with a laptop and a potted plant.

**Q**: what is the green logo on the car?

**A**: The green logo on the car is the Monster Energy logo.

**Q**: Introduce the player on the right.

**A**: The player on the right is named Kaká, and he is wearing a jersey with the number 8.

**Q**: Is there any reflection of zebra in water?

**A**: Yes, there is a reflection of the zebra in the water.

**Q**: Can you give a short introduction to this painting?

**A**: The painting is a famous artwork by Vincent Van Gogh, featuring a night scene with a sky filled with stars. The scene is set in a village, with a church steeple visible in the distance. The painting captures the beauty of the night sky and the tranquility of the village, creating a serene atmosphere. The combination of the stars, the church steeple, and the village buildings make this painting a captivating representation of the artist's vision.

**Q**: Can you give a short introduction to this person?

**A**: The image features a statue of an old man with a long beard, sitting and holding a staff in his hands. The statue appears to be a representation of a wise or respected figure, possibly a religious leader or a historical figure. The statue is situated in front of a building, possibly a temple or a museum, and it serves as a symbol of wisdom, knowledge, or spiritual guidance.

**Q**: Provide a detailed description of the given image.

**A**: The image features a dining table with a white plate filled with a variety of breakfast foods. The plate contains eggs, toast, and a selection of fruits, including strawberries and blueberries. There are also sausages on the plate, adding to the delicious assortment of breakfast items. A cup of coffee is placed next to the plate, completing the breakfast setup. A fork is positioned on the table, ready for use. The table setting is inviting and perfect for a morning meal.

**Q**: Can you give a short introduction to this movie?

**A**: The movie "The Dark Knight" is a 2008 superhero film directed by Christopher Nolan. It is the second installment in the Batman film series and stars Heath Ledger as the Joker, along with Christian Bale as Batman/Bruce Wayne, Michael Caine as Alfred, and Maggie Gyllenhall as Rachel Dawes. The film follows Batman as he battles the Joker, a psychopathic criminal mastermind who seeks to take over Gotham City. The movie is known for its intense action sequences, complex plot, and the iconic performance of Heath Ledger as the Joker.

Figure 3. Examples of the generated responses using the self-speculative decoding (SSD) on MM-Vet [22], with accepted tokens by the target model being highlighted in green.

## D. Evaluation Benchmarks

In this section, we provide a brief overview of the benchmarks used in our experiments.

**GQA** [7] is a benchmark that focuses on visual scene understanding and reasoning, leveraging scene graphs, questions, and images. It incorporates spatial relationships and object properties, posing challenges for models to perform accurate visual reason under complex visual environments.

**MMBench** [15] adopts a hierarchical evaluation approach with three levels: Level-1 (perception and reasoning), Level-2 (six sub-abilities), and Level-3 (20 specific dimensions). This structured framework allows for a comprehensive evaluation of model performance, making it an effective tool for assessing a wide range of visual understanding capabilities. We denote it as “MMB” in the main text.

**MME** [4] assesses models across 14 subtasks that probe both perceptual and cognitive skills. Carefully crafted instruction-answer pairs guarantee a fair and comprehensive evaluation of a model’s multimodal performance. The final score reported on this benchmark is the summation of both the perception and cognition scores.

**ScienceQA** [16] spans multiple scientific fields, including natural, language, and social sciences, with questions organized into 26 topics, 127 categories, and 379 skills. It evaluates a model’s multimodal comprehension, multi-step reasoning, and interpretability, providing a rich testbed for assessing scientific knowledge application in visual contexts. In our experiments, we only evaluate the performance on the samples with images, denoted as “$\text{SQA}^{\text{I}}$” in the experimental tables.

**VQA-v2** [5] is a large-scale benchmark featuring 265K images of real-world scenes and objects, with each image paired with open-ended questions and 10 human-provided ground truth answers.

**TextVQA** [18] tests a model’s ability to process and reason about text embedded within images. By requiring the integration of visual and textual information, it serves as a critical benchmark for evaluating text-based reasoning in visual contexts. To save space, we denote it as “$\text{VQA}^{\text{T}}$” in the experimental tables.

**POPE** [10] targets object hallucination evaluation by posing binary questions about object presence in images. It employs metrics such as Accuracy, Recall, Precision, and F1 score across three sampling methods. The reported score is calculated by the mean accuracy over the three indicators: adversarial, random, and popular.

**MMMU** [24] challenges models with tasks requiring college-level expertise and reasoning skills. It comprises 11.5K questions drawn from exams, quizzes, and textbooks, spanning six key disciplines: Art & Design, Business, Science, Health & Medicine, Humanities & Social Science, and Tech & Engineering. Featuring 30 subjects and 183 sub-fields, MMMU involves diverse image types, *e.g.*, charts, diagrams, and chemical structures, demanding advanced perceptual and domain-specific reasoning abilities akin to those of human experts.

**MM-Vet** [22] evaluates six fundamental vision-language capabilities: recognition, OCR, knowledge, language generation, spatial awareness, and mathematical reasoning. It examines 16 specific combinations of these skills through quantitative metrics, offering a nuanced perspective on a model’s proficiency in tackling intricate multimodal tasks.

**TGIF-QA** [8] adapts image question answering to the video domain, specifically targeting GIFs. With 165K question-answer pairs, it introduces tasks such as counting repetitions, identifying repeating actions, detecting state transitions, and frame-specific QA. These tasks demand detailed spatio-temporal analysis, making it a rigorous test for video comprehension. We employ GPT-3.5-turbo to assist in the evaluation of accuracy (same for the following three benchmarks). We denote it as “TGIF” in the main text.

**MSVD-QA** [20] is built on the Microsoft Research Video Description (MSVD) dataset [1], which includes 1,970 video clips and about 50,500 QA pairs. Its open-ended questions span five types—what, who, how, when, and where—offering a diverse and widely used evaluation for video question answering and captioning tasks. We denote it as “MSVD” in the main text.

**MSRVTT-QA** [1] comprises 10K video clips and 243K QA pairs, presenting a complex challenge due to the need to process both visual and temporal information. Like MSVD-QA, it features five question types, testing a model’s ability to understand and reason about dynamic video content. We denote it as “MSRVTT” in the main text.

**ActivityNet-QA.** ActivityNet-QA [23] offers 58K human-annotated question-answer pairs across 5.8K videos from the ActivityNet dataset. Its questions include motion, spatial relationships, and temporal dynamics, requiring long-term spatio-temporal reasoning and making it an popular benchmark for evaluating advanced video understanding capabilities. We denote it as “ActivityNet” in the main text.

## References

[1] David L. Chen and William B. Dolan. Collecting highly parallel data for paraphrase evaluation. In *ACL*, Portland, OR, 2011. 7

[2] 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. In *European Conference on Computer Vision*, pages 19–35. Springer, 2024. 3, 4, 5

[3] Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, et al. Layerskip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*, 2024. 2

[4] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Zhenyu Qiu, Wei Lin, Jinrui Yang, Xiawu Zheng, et al. Mme: A comprehensive evaluation benchmark for multimodal large language models. *arXiv preprint arXiv:2306.13394*, 2023. 7

[5] 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/CVF Conference on Computer Vision and Pattern Recognition*, pages 6904–6913, 2017. 7

[6] Jarvis Guo, Tuney Zheng, Yuelin Bai, Bo Li, Yubo Wang, King Zhu, Yizhi Li, Graham Neubig, Wenhu Chen, and Xiang Yue. Mammoth-vl: Eliciting multimodal reasoning with instruction tuning at scale, 2024. 2

[7] 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*, 2019. 5, 7

[8] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Tgif-qa: Toward spatio-temporal reasoning in visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2758–2766, 2017. 7

[9] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023. 2

[10] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. *arXiv:2305.10355*, 2023. 7

[11] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. 1

[12] Fangcheng Liu, Yehui Tang, Zhenhua Liu, Yunsheng Ni, Duyu Tang, Kai Han, and Yunhe Wang. Kangaroo: Lossless self-speculative decoding for accelerating llms via double early exiting. *Advances in Neural Information Processing Systems*, 37:11946–11965, 2025. 2

[13] 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

[14] Ting Liu, Liangtao Shi, Richang Hong, Yue Hu, Quanjun Yin, and Linfeng Zhang. Multi-stage vision token dropping: Towards efficient multimodal large language model. *arXiv preprint arXiv:2411.10803*, 2024. 3

[15] 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. 7

[16] 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. 7

[17] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024)*, 2024. 1

[18] 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. 5, 7

[19] 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. 3

[20] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *Proceedings of the ACM international conference on Multimedia*, pages 1645–1653, 2017. 7

[21] Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. Visionzip: Longer is better but not necessary in vision language models. *arXiv preprint arXiv:2412.04467*, 2024. 3, 4, 5

[22] Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. Mm-vet: Evaluating large multimodal models for integrated capabilities. *arXiv preprint arXiv:2308.02490*, 2023. 6, 7

[23] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *AAAI*, pages 9127–9134, 2019. 7

[24] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9556–9567, 2024. 7

[25] Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [cls] attention is all you need for training-free visual token pruning: Make vlm inference faster. *arXiv preprint arXiv:2412.01818*, 2024. 3

[26] 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. 3