Appendix Contents

A. Experiment Details

A.1. Workflow of the Prompt Highlighter

For a more comprehensive understanding of the method part of Prompt Highlighter, we provide pseudo-code algorithmic workflows in Algorithm 1 and Algorithm 2. In this context, the attention activation outlined in Algorithm 2 acts as the modified forward function in all Self-Attention layers during the multi-modal LLMs' inference $\tilde{\mathbf{P}}_\Theta(x_\eta|\mathbf{s}, \mathbf{m})$, as discussed in Sec. 3.2 and illustrated in Algorithm 1.

Algorithm 1 Highlighted Guidance Control

Require: Pre-trained LLM or VLM decoder $\mathbf{P}_\Theta$, input multi-modal tokens $\mathbf{x}$, binary highlight mask $\mathbf{m}$, guidance strength γ, and scaling factor α, δ

Initialize token sequence $\mathbf{x} = \{x_1, \ldots, x_N\}$
Initialize conditional embedding $\mathbf{s} = f(\mathbf{x})$ and unconditional embedding $\bar{\mathbf{s}}$ as $\bar{\mathbf{s}} = (\alpha - 1)\mathbf{m} \cdot f(\mathbf{x}) + f(\mathbf{x})$ (Eq. (5)).
Initialize current token index $\eta = N$, output token sequence $\mathbf{y} = [\,]$

while x_η is not </s> **do**
  Calculate the greedy decoded prediction x_η in Eq. (6):
  $$x_{\eta+1} = \text{argmax}(\gamma \log \bar{\mathbf{P}}_\Theta(x_{\eta+1}|\mathbf{s}, \mathbf{m}) - (\gamma - 1) \log \bar{\mathbf{P}}_\Theta(x_{\eta+1}|\bar{\mathbf{s}}, -\delta\mathbf{m}))$$
  Update output sequence:
  $\mathbf{y}$.append($\mathbf{P}_\Theta$.tokenizer.decode($x_{\eta+1}$))
  Update next token embedding and mask:
  $s_{\eta+1} = \bar{s}_{\eta+1} = f(x_{\eta+1}), m_{\eta+1} = 0, \eta = \eta + 1$
end while
return Output sequence $\mathbf{y}$

Algorithm 2 Attention Activation in Self-Attention Layer. For simplicity, we demonstrate with single-head attention.

Require: Input hidden state $\mathbf{q} = \{q_1, \ldots, q_{\eta-1}\}$ and vanilla attention mask $\mathbf{m}_{\text{attn}}$, highlight mask $\mathbf{m}$, Scaling factor β, Self-Attention Module $M_{\text{SA}} \subset \tilde{\mathbf{P}}_\Theta$.

$(Q, K, V) = M_{\text{SA}}$.proj_qkv($\mathbf{q}$)
Calculate the attention score map:
$\mathbf{k} = Q@K^\top + \mathbf{m}_{\text{attn}}$
Initialized the highlighted attention mask:
$\mathbf{w} = \mathbf{m}$.expand_as($\mathbf{k}$)
Activate score via attention map (Eqs. (7) and (9)):
$\mathbf{h} = \log(\beta)\mathbf{w} + \mathbf{k}$
Then complete the remaining operations in self-attention:
$\mathbf{p} = \text{softmax}(\mathbf{h})$
return $\mathbf{q}_{\text{out}} = M_{\text{SA}}$.proj_out($\mathbf{p}@V$)

A.2. Quantitative Evaluation

A.2.1 VLM Benchmarks

We use the same prompts and test scripts for our benchmark tests as those in the LLaVA-v1.5 codebase [23]. The only difference is our highlighting of the image context in the inputs (*i.e.*, $m_i = 1$ when $s_i \in \mathbf{s}^{\text{im}}$). We adopt softmax, instead of log softmax used in the multiple-token prediction tasks, as the token probability rescale function. As discussed in Sec. 4.4, we observe slight variations in results across different test sets based on the parameters of attention activation. In Tab. 7, we showcase the impact of changes in the β value on the benchmark tests. Our method still achieves the second rank on the current MMBench-`dev` leaderboard, even when using a uniform parameter of $\beta = 2.0$.

We adopt the MME and MMBench datasets because they offer comprehensive evaluations of VLMs across multiple dimensions and feature over 10K of VQA questions. This makes them ideal for assessing the overall competency of Vision-Language Models. We present a detailed evaluation result of MMBench in Tab. 6. With the same pre-trained weight as LLaVA-v1.5, our training-free method consistently outperforms the baseline across nearly all evaluation dimensions. Given that our approach is compatible with VLM frameworks that use token-level embedding input and a Transformer-based language decoder, it positions the Prompt Highlighter as a **training-free plugin** for integrating context-highlighting abilities into these models. We further evaluate the MMBench benchmark using the current state-of-the-art model, InternLM-VLComposer [32]. Our method demonstrated a performance improvement, increasing the SOTA performance from 74.8 to 75.3 in the `dev` split with this representative Q-Former-based approach. These results suggest that though our method is designed to create new interactive ways instead of focusing on bench-

method	MMBench-dev							MMBench-test						
	Overall	LR	AR	RR	FP-S	FP-C	CP	**Overall**	LR	AR	RR	FP-S	FP-C	CP
MMICL [50]	67.9	**49.2**	71.6	**73.0**	66.7	57.2	77.2	65.2	**44.3**	**77.9**	**64.8**	66.5	53.6	70.6
mPLUG-Owl2 [14]	66.5	32.2	**72.4**	60.9	68.6	60.1	79.4	66.0	43.4	76.0	62.1	68.6	55.9	73.0
Sphinx [51]	67.2	33.1	67.3	58.3	**74.4**	**59.4**	80.7	67.5	32.9	73.6	57.8	72.1	**63.2**	**79.2**
LLaVA-v1.5 [23]	67.7	41.7	69.7	63.5	70.0	59.3	80.2	67.0	39.9	74.7	61.6	70.9	59.9	75.4
+ Prompt Highlighter	**69.7**	44.2	70.6	68.7	73.7	59.3	**80.9**	**69.5**	42.6	77.5	64.3	**75.0**	62.0	76.4
Improvement	+2.0	+2.5	+0.9	+5.2	+3.7	0.0	+0.7	+2.5	+2.7	+2.8	+2.7	+4.1	+2.1	+1.0

Table 6. Detailed comparison in MMBench-dev and MMBench-test[26]. The categories include Logic Reasoning (LR), Attribute Reasoning (AR), Relation Reasoning (RR), Instance-Level Fine-Grained Perception (FP-S), Cross-Instance Fine-Grained Perception (FP-C), and Coarse Perception (CP). The improvement of our method over the LLaVA-v1.5 [23] baseline is reported in the last row for each category. We highlight **the best** and underline the second best result for each column.

method	MME	$\text{MMB}_{\texttt{dev}}$	$\text{MMB}_{\texttt{test}}$
LLaVA-v1.5 [23]	1531.3	67.7	67.0
PH (0.01, **3.0**, 1.3)	1537.3	**69.7**	68.7
PH (0.01, **2.0**, 1.3)	**1552.5**	68.7	**69.5**
PH (submitted)	**1552.5**	**70.1**	**70.7**

Table 7. Hyper-parameter settings in VLM benchmarks. Evaluations are conducted on MME Perception [27] and MMBench (MMB) [26]. We list a combination of hyper-parameters (α, β, γ) for our method (PH for Prompt Highlighter).

sub-task	baseline	preference
Text Partial Highlighter	Vicuna-13B	70.6%
Image Partial Highlighter	LLaVA	74.5%
Image Understanding	InstructBLIP	76.5%
Image Caption	LLaVA-v1.5	80.4%
Description → Image	LLaVA-v1.5 + DALLE-3	84.3%
Overall	-	**77.3%**

Table 8. A fine-grained user study result.

mark improvement, it still proves to be competitive in general applications.

A.2.2 Reliable Descriptions

As discussed in Sec. 4.4, we observe a trade-off between the attention concentration and QA performance that appeared in the benchmarks. Therefore, compared with experiments in VLM benchmarks, we adopt a relatively more aggressive parameter setting in the MSCOCO caption experiment: $(\alpha, \beta, \gamma) = (0.01, 7.0, 2.0)$. We utilize a CLIP ViT-Base-32 model [30, 52] to test the CLIP Score [48] and to extract embeddings for both text → image and image → text retrieval metrics. Our experiment uses a simple input prompt, `'Describe this image.'`. In the comparison in Tab. 2, we report the official result reported by CoCa-CFG [22] with the same γ value of 2.0[1]. As CLIP has a maximum token length limitation of 77, we split captions at semicolons and periods and input the longest possible concatenated sequence to circumvent this limitation. For all text-to-image generation results in figures, we query the generated text description to the Bing Image Creator powered by DALLE-3 [44] [`www.bing.com/images/create`].

[1]CoCa-CFG achieves caption → image retrieval recall with R@1=49.4 and R@5=75.7 with $\gamma = 3.0$, while it remains the same S-CLIP=0.808.

A.2.3 User Study

We present the detailed results of the user study in Tab. 8. Using different sub-task examples, we engaged various baseline models for different tasks to demonstrate the consistent preference for Prompt Highlighter across a broad array of frameworks and tasks. The user preference results affirm that our method delivers superior performance and flexibility, consistently outperforming the baseline models across a wide range of tasks.

A.3. Attention Map Visualization

In this section, we delve into the attention map visualization, expanding on the verification experiments discussed in Sec. 4.5. In Fig. 10, we illustrate an example akin to Fig. 3, demonstrating the persistence of the band-like pattern in different layer segments in the Vicuna-13B model [8]. The band-like property is clear and consistently evident across various layers. Given that these band-like token activation patterns are distributed across different layers, we simply incorporate the attention activation modification in all self-attention layers. Furthermore, the attention maps referenced in Fig. 9 to calculate gradients and attention contributions are computed based on the averaged attention map across layers and heads.

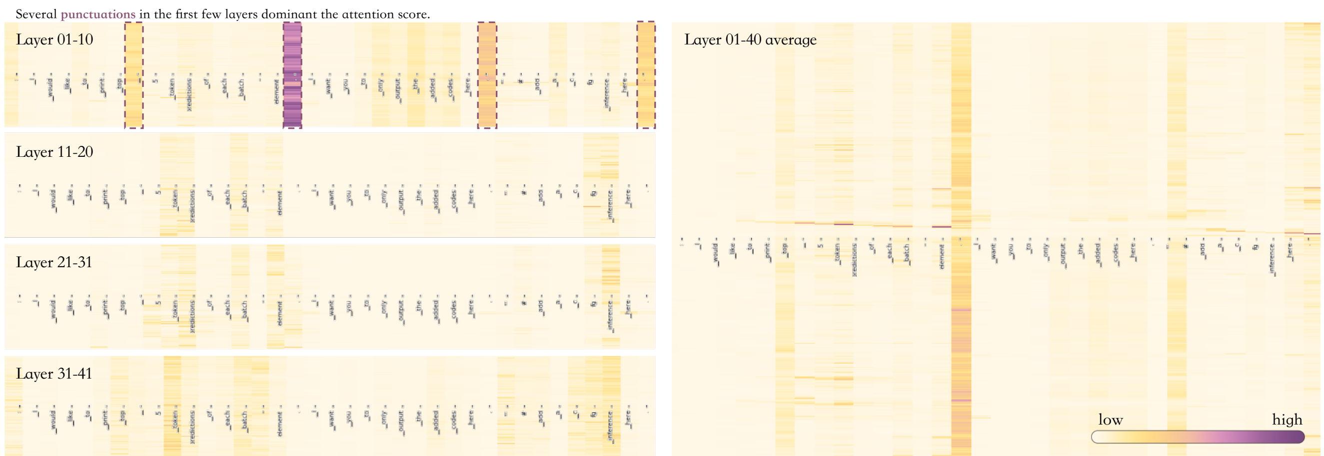


Figure 10. An additional example of attention maps across different layer segments (***left***) and the averaged visualization with more generated tokens (***right***). We mark tokens representing punctuation, which draw significant attention in the initial layers. Beyond these tokens, a consistent correlation between tokens forms a band-like pattern that can be observed across all layers.

Figure 11. An example illustrating the influence of different base models. Given the same highlighted context, the output of the model may be constrained by the capabilities of the base model.

A.4. Limitation Analysis

A.4.1 Speed and Memory

For processing two tokens compared to single-beam (greedy) inference in image description, batched inference

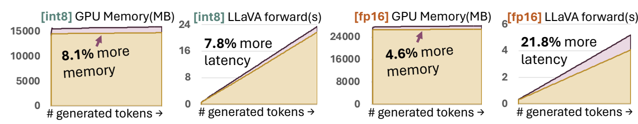


Figure 12. Speed and memory overhead analysis. We evaluate VLM inference both in int8 and fp16 precision.

enables parallel computation, adding only 7% and 10% to the memory and speed costs, respectively. We corroborate this with plotted relation curves for processing $N = 0, \ldots, 140$ tokens, demonstrating that our method maintains controllability with minimal extra costs.

As discussed in Sec. 4.5, Prompt Highlighter imposes an additional computational load on the unconditional branch, and we validate its tangible impact in Tab. 5. Memory consumption with 8-bit loading confirms that the Prompt Highlighter is designed to be memory-efficient, making it feasible to run inference on a 13B model using commonly available GPUs, such as the NVIDIA-3090. This added load becomes noticeable when evaluating single-token prediction tasks, as the model is required to compute double KV values among tokens in its initial inference step. However, when employing KV-cache in the multiple-token generation, this extra load can be alleviated, as the doubled calculation with cached KVs does not become the bottleneck for inference speed or GPU memory consumption.

For processing two tokens compared to single-beam (greedy) inference in image description, batched inference enables parallel computation, adding only 7% and 10% to the memory and speed costs, respectively. We corroborate this with plotted relation curves in Fig. 12 for processing $N = 0, \ldots, 140$ tokens, demonstrating that our method maintains controllability with minimal extra costs. We also

User: Please give me a plan to learn about computer vision.

[Vicuna] **Normal Inference:**
1. Start with the basics··· 2. Learn about image processing···
3. Learn a programming language··· 4. Learn about OpenCV···
5. Learn about deep learning··· 6. Practice with projects···
7. Read books and articles··· 8. Join online communities···

LLM-CFG:
1. Learn the basics of programming ··· 2. Learn the basics of image processing ··· 3. Learn about OpenCV ··· 4. Learn about machine learning ··· 5. Learn about computer vision algorithms ··· 6. Practice with datasets ··· 7. Build projects ··· 8. Stay up to date ···

Prompt Highlighter:
1. Foundations of Computer Vision··· 2. Image Processing···
3. Feature Detection··· 4. Object Detection··· 5. Image Segmentation··· 6. Deep Learning for Computer Vision···

Figure 13. One example of the comparison with LLM-CFG [21].

[Vicuna] Please give me a plan to learn about computer vision.

$\alpha > 0.1$ Sure, here's a plan to help you learn computer vision···

$\alpha = 0.01$ Certainly! Here's a plan to help you learn Python···

LLM-CFG Certainly! Here's a plan to help you learn ASSTST···

Figure 14. Output example for the unconditional branch with different unconditional context designs.

confirm that the commonly utilized inference strategy of introducing an extra beam search branch does not impair the quality of the descriptive task (S-CLIP 0.809→0.807 in Tab. 5), despite imposing an additional computational load. This also illustrates the practical effectiveness of Prompt Highlighter.

A.4.2 Constrained by the Base-Model

Our method stems from the base model and its plug-and-play nature. As illustrated in Fig. 11, the base model InstructBLIP [15] is not adept at dialog generation. Consequently, with a highlighted prompt context, despite the output text being more related to the highlighted part, it fails to generate a dialogue effectively. This suggests that while the Prompt Highlighter can augment existing abilities in pre-trained multi-modal LLMs, it cannot endow them with new capabilities.

B. Additional Disccusions

B.1. Compare with LLM-CFG

The comparison between Prompt Highlighter and LLM-CFG [21] is visually demonstrated in Fig. 13. Our method

User: <marked_image> Describe this image, emphasize the marked area.

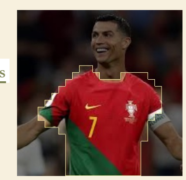
Normal Inference (add marks and arrow): The image features a man wearing a red and green soccer jersey, smiling and posing for the camera. He is wearing a wristband on his left wrist, and his shirt is numbered with the number seven. The man is standing in a room, possibly a locker room, with a few other people in the background.
The main focus of the image is the man in the red and green soccer jersey, and a red arrow is pointing towards him, drawing attention to his presence in the scene.

User: <original_image> Describe this image, and emphasize more on the jersey

Normal Inference (prompt engineering): The image features a man wearing a red and green soccer jersey, smiling and posing for the camera. The jersey has a distinctive design, with a combination of red and green colors, and the number seven displayed on it. The man appears to be a soccer player, possibly a professional athlete, as he is wearing a sports uniform and is in a stadium setting.

User: <masked_image> Describe this image

Normal Inference (physically "highlight" the jersey part and input a masked image):
The image features a man wearing a red and green soccer jersey, standing on a field and smiling. He appears to be a soccer player, possibly a professional athlete, as he is wearing a uniform and is in a stadium setting. The man is the main focus of the image, and he is surrounded by a crowd of people, indicating that he might be participating in a game or event.



User: < original _image, **m**> Describe this image

Prompt Highlighter: The image features a soccer player wearing a red, green, and gold jersey with the number 7 on it. The player is smiling and appears to be enjoying the game. He is wearing green and red stripes on his jersey, which are visible on the sleeves and the green stripe on the side. The player is also wearing a green and white armband on his right arm. The scene captures the excitement and enthusiasm of the soccer player as he participates in the game.

Figure 15. A comparison of Prompt Highlighter with different vision-language prompt manipulation approaches.

stands out by offering a more granular control mechanism that allows specific tokens within the context to be highlighted. Conversely, LLM-CFG, with its basic prompt-level differentiation and without a mechanism to directly manipulate model feature interactions (*e.g.*, our attention activation strategy), often generates outputs that bear noticeable similarity to the original, regardless of substantial γ increases.

Using the identical prompt as in Fig. 13, we consolidate this claim by exhibiting the independent inference outcomes of the unconditional branch in Fig. 14. In addition, we demonstrate the significance of assigning the α value at the embedding layer. As shown in Fig. 14 and as per our discussion in Sec. 3.1, it becomes apparent that a small scalar multiplication α in the embedding space leaves the overall contextual understanding of the unconditional branch largely unaffected. Such a fine-grained difference assists in making more distinguishing token choices during the inference process in Eq. (6). In contrast, LLM-CFG's unconditional branch omits part of the prompt, resulting in an obstacle to generating results that effectively underscore the highlighted sections.

B.2. Compare with other Highlight Methods

To demonstrate the non-trivial nature of our method, we evaluated it alongside several intuitive prompt engineering approaches, which include adding adjectives, capitalizing text, and explicitly marking in images. Comparative examples in LLM and VLM are provided in Figs. 1 and 15, respectively. Our observations suggest that the employment of additional prompts or explicit emphasis can occasionally lead to unpredictable results. This is primarily due to the introduction of extra information, while models may not truly emphasize correlated parts. In contrast, our method seamlessly accommodates all context information and produces focused text outputs. This demonstrates that the design of our method not only offers a high degree of flexibility but also enables the generation of more contextually appropriate outputs.

B.3. Highlighting the Whole Image.

We demonstrate in Algorithm 2 and Sec. 3.2 that self-generated tokens may lead to model hallucinations. The model's attention is steered towards trustworthy content by purposefully highlighting the reliable input context, such as the entire image, mitigating hallucinations in extended text generation. An example from MME is provided below to show how this approach effectively curbs hallucinations during evaluations.

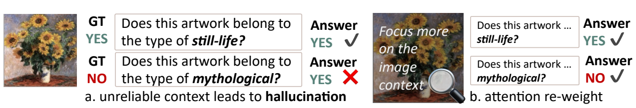


Figure 19. An example of highlighting image context to eliminate hallucination.

C. Showcases

C.1. More Visual Results

Additional results of Prompt Highlighter in LLM and VLM are provided in Fig. 16 and Fig. 17, respectively.

In the first example of Fig. 16, we showcase the primary part of the output of the example in Fig. 6. Our results concentrate more on the highlighted section, offering professional reference books to support user learning "computer vision". The second example exhibits a case for a multi-segment mask input, where the model is steered to output code snippets at the appropriate positions by highlighting the requirement part of the text and the comment as the position identifier. Fig. 17 demonstrates a range of applications and capabilities of Prompt Highlighter in VLM from various perspectives, including image content comprehension (case 1), focused generation (case 2), improved image descriptions (case 3), and explicit generation control (case 5). These instances underscore the performance enhancement and customizable generation control facilitated by Prompt Highlighter. We also present some challenging cases due to the limited perceptual capacity of the base model.

Moreover, we present results of text-to-image generation based on the text descriptions generated by our method in Fig. 18. These outcomes vouch for the superior capability of our approach in descriptive tasks and its proficiency in generating higher-quality image-text matching data.

C.2. Multi-Round Interactive Conversation

Up to this point, the examples and experimental scenarios we've showcased are for single-round dialogues or QAs. As a general-purpose interactive inference pipeline, our method can also support multi-round interactions and conversations. We provide a schematic diagram of multi-round conversation at the top of Fig. 20. The multi-round conversations' generation can be decomposed into single-round generations with continuously growing context, allowing Prompt Highlighter to change the focus of user input in each round by updating the input mask $\mathbf{m}$ and resetting previous KV caches.

Below the pipeline, we demonstrate how to resolve the image understanding problem, presented in Fig. 17 case 4, through multi-round conversation and human interactions. By highlighting multi-modal tokens and guiding the model to understand the image content in a segmented manner, users can "teach" the model to solve problems that it was previously unable to solve.

Figures 15-18 are presented on the following pages ↓

Case1: A full version of the first example in Fig. 6 Normal Inference Prompt Highlighter

User: Please give me a plan to learn about computer vision.

1. Start with the basics: Before diving into the advanced concepts of computer vision, it's essential to have a solid understanding of the fundamentals. Learn about the different types of computer vision, such as object recognition, image segmentation, and motion analysis.
2. Learn about image processing: Image processing is a crucial aspect of computer vision. Learn about image filtering, edge detection, and other image processing techniques.
3. Learn a programming language: Python is the most popular language used in computer vision. Learn the basics of Python and familiarize yourself with the libraries commonly used in computer vision, such as OpenCV, TensorFlow, and Keras.
4. Learn about OpenCV: OpenCV is a powerful library for computer vision tasks. Learn about its various functions and how to use them to implement computer vision algorithms.
5. Learn about deep learning: Deep learning is a subset of machine learning that uses neural networks to perform tasks. Learn about the basics of deep learning and how it's used in computer vision.
6. Practice with projects: The best way to learn computer vision is by practicing with projects. Start with simple projects and work your way up to more complex ones. This will help you apply what you've learned and improve your skills.
7. Read books and articles: There are many books and articles available on computer vision. Read them to learn about the latest developments in the field and to gain a deeper understanding of the concepts.
8. Join online communities: Join online communities, such as forums and social media groups, to connect with other computer vision enthusiasts. This will help you learn from others and stay up-to-date with the latest developments in the field.

1. Foundations of Computer Vision: Start by learning the basics of computer vision, including image processing, feature detection, and image matching. Read books like "Computer Vision: Algorithms and Applications" by Richard Szeliski and "Introduction to Computer Vision" by Richard Szeliski, et al.
2. Image Processing: Learn image processing techniques such as filtering, thresholding, and edge detection. Read books like "Image Processing: The Fundamentals" by John C. Bovik and "Digital Image Processing" by Rafael C. Gonzalez and Richard E. Woods.
3. Feature Detection: Learn about feature detection techniques such as SIFT, SURF, and ORB. Read papers like "SIFT: Scale-Invariant Feature Transform" by David G. Lowe and "ORB: Oriented FAST and Rotated BRIEF SIFT" by Yi Liu and Aki Kanade.
4. Object Detection: Learn about object detection techniques such as Haar cascades, HOG, and Viola-Jones algorithms. Read papers like "Haar Cascades for Real-Time Object Detection" by Andreas Santner, et al. and "A Survey of Object Detection" by Paul Viola and Michael Jones.
5. Image Segmentation: Learn about image segmentation techniques such as connected component analysis, region growing, and watershed segmentation. Read books like "Image Segmentation: Algorithms and Applications" by T. P. Singh and "Computer Vision: A Modern Approach" by David A. Forsyth and Jean Ponce.
6. Deep Learning for Computer Vision: Learn about deep learning techniques for computer vision, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Read books like "Deep Learning for Computer Vision" by Rajalingapuram Kannan and "Computer Vision with Deep Learning" by S. S. K. Tiwari and S. C. Gupta.
7. Practice and Projects: Practice your computer vision skills by working on projects and datasets such as ImageNet, OpenCV, and Kaggle.

Case2: Specific requirement in code generation. We partial highlight both the text prompt and the position comment in the code segment.

```python
# add a cfg inference here.
def CFG_greedy_forward(input, model, guidance=1.5, max_length=256):
    new_ids = [1]
    for i in tqdm(range(max_length)):
        if i == 0: # update the augmented key-values
            dict_outputs = model(inputs_embeds=input, past_key_values=key_values, return_dict=True)
        else:
            dict_outputs = model(inputs_embeds=input, return_dict=True, use_cache=True, output_attentions=True)

        last_logit = torch.argmax(dict_outputs.logits[:, -1:], dim=-1)
        # INSERT CODE HERE
        new_ids += last_logit
        if last_logit == 2: # stop token.
            break
        last_logit_embed = model.get_input_embeddings()(last_logit)
        input = torch.cat((input, last_logit_embed), dim=1)
        cfg_batched_input = torch.cat([embed, highlighted_embed])
```

User: I would like to print top 5 token predictions of last_logit. Directly output the newly added codes.

You can simply print the top 5 token predictions of the last logit by sorting the logits in descending order and taking the first 5 elements. Here's the updated code:

```python
# add a cfg inference here.
def CFG_greedy_forward(input, model, guidance=1.5,max_length=256):
    new_ids = [1]
    for i in tqdm(range(max_length)):
        if i == 0: # update the augmented key-values
            dict_outputs = model(inputs_embeds=input, past_key_values=key_values, return_dict=True)
        else:
            dict_outputs = model(inputs_embeds=input, return_dict=True, use_cache=True, output_attentions=True)
        last_logit = torch.argmax(dict_outputs.logits[:, -1:], dim=-1)
        # INSERT CODE HERE
        new_ids += last_logit
        if last_logit == 2: # stop token.
            break
        last_logit_embed = model.get_input_embeddings()(last_logit)
        input = torch.cat((input, last_logit_embed), dim=1)
        cfg_batched_input = torch.cat([embed, highlighted_embed])
        return cfg_batched_input

    # sort the logits in descending order and take the first 5 elements
    logits = dict_outputs.logits[:, -1:]
    sorted_logits = logits.sort(desc=True)[:5]

    # print the top 5 token predictions
    print(sorted_logits)
```

Here's the added part to print the top 5 token predictions of the last logit:

```python
# INSERT CODE HERE
top_tokens = sorted(last_logit, key=lambda x: dict_outputs.logits[0, x], reverse=True)[:5]
print(top_tokens)
```

This code sorts the last logit based on the logits and selects the top 5 tokens. The logits represent the model's confidence in each token being the next one in the sequence, so the top 5 tokens with the highest logits are likely to be the correct predictions. Finally, the code prints the added part.

Figure 16. Two more examples in pure text partial highlighter. A Vicuna-13B-v1.1 [8] is used as the base model.

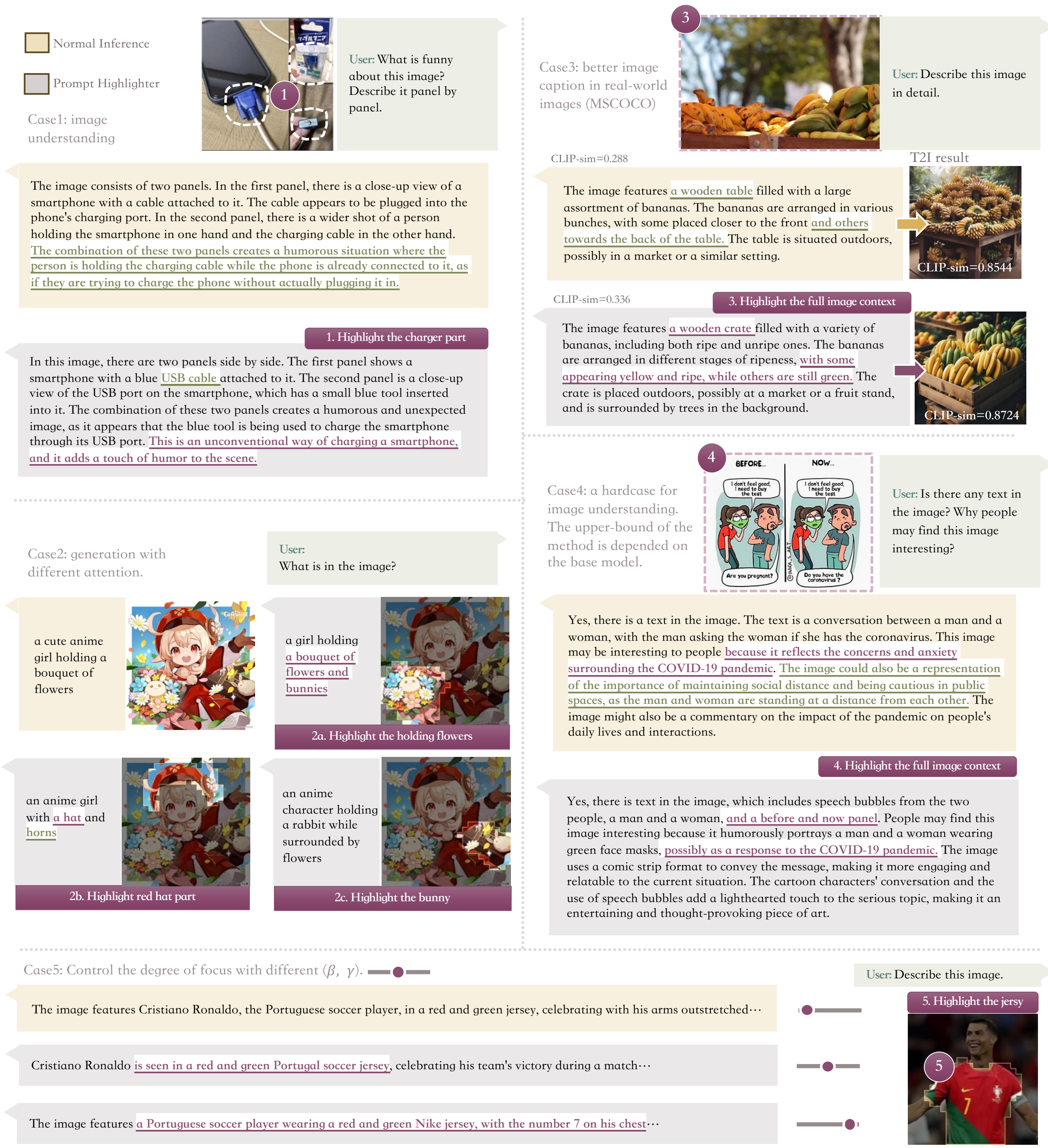


Figure 17. More results with multi-modal inputs. We demonstrate three applications (1,2,3,5) and one hard case (4) for image understanding. We use InstructBLIP-Vicuna-13B [15] as the base model of cases 1,2,5 and LLaVA-v1.5 [23] 13B as the base model of cases 3,4.

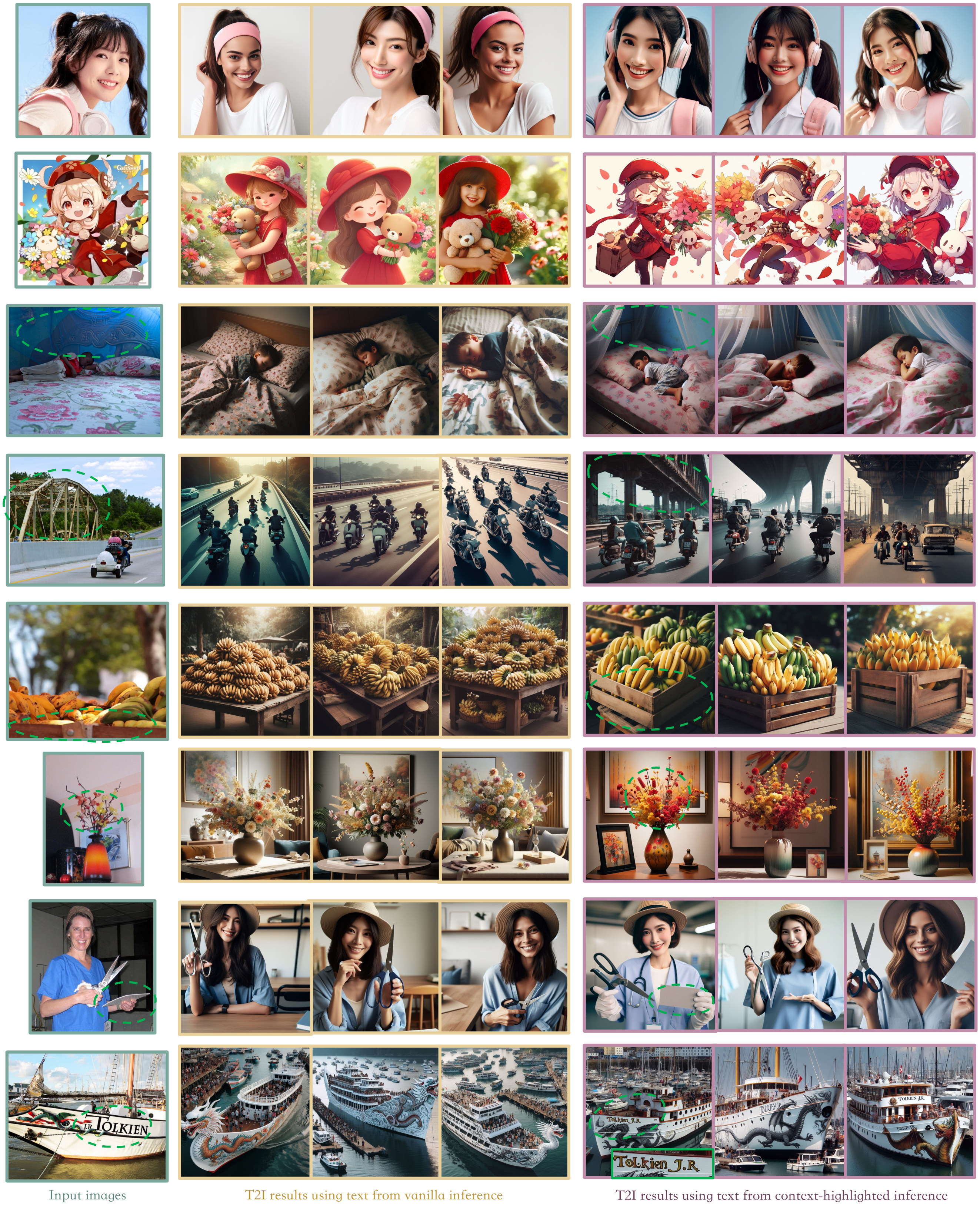


Figure 18. More text-to-image examples with image generation by DALLE-3 [44]. We have marked areas in some figures that generate better results due to different captions.

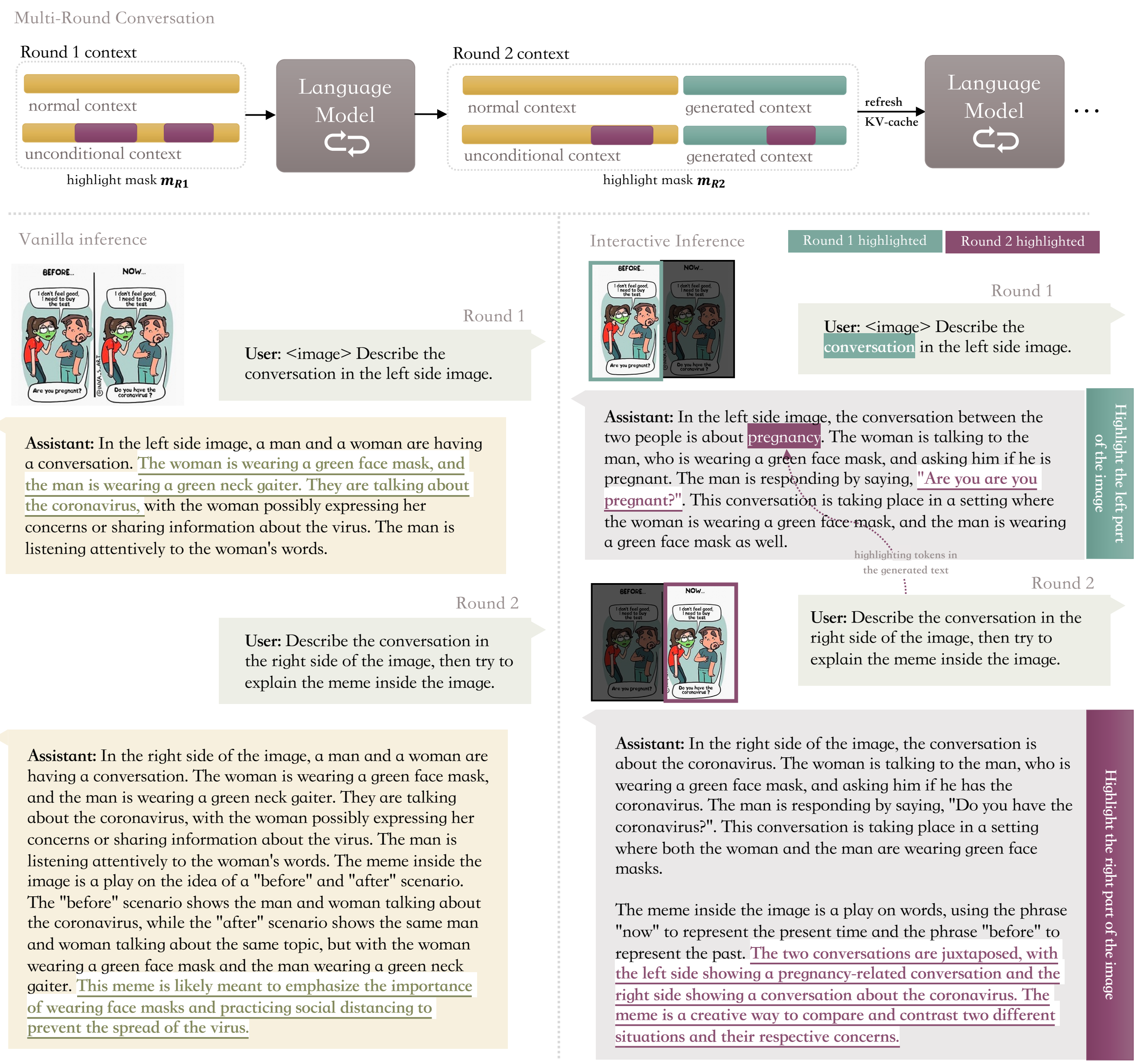


Figure 20. Multi-round interactive conversation powered by Prompt Highlighter. We illustrate the multi-round interactive conversation pipeline on the ***top***. We provide a comparison in a multi-round conversation between the vanilla inference (***left***) and Prompt Highlighter (***right***). In this example, the user highlights different contexts in two rounds.

References

[1] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018. 1, 2

[2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[3] 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. *NeurIPS*, 33:1877–1901, 2020.

[4] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*, 2019.

[7] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 2, 4, 5

[8] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2023. 4, 5, 10, 14

[9] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022. 1, 2

[10] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 1, 2, 4, 5

[11] 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. 2, 4

[12] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *ICML*, pages 12888–12900. PMLR, 2022.

[13] 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. 2, 4

[14] Qinghao Ye, Haiyang Xu, Guohai Xu, Jiabo Ye, Ming Yan, Yiyang Zhou, Junyang Wang, Anwen Hu, Pengcheng Shi, Yaya Shi, et al. mplug-owl: Modularization empowers large language models with multimodality. *arXiv preprint arXiv:2304.14178*, 2023. 4, 7, 10

[15] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023. 1, 2, 5, 12, 15

[16] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023. 1, 2

[17] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *NeurIPS*, 35:24824–24837, 2022. 2

[18] Significant-Gravitas. Significant-gravitas/autogpt: An experimental open-source attempt to make gpt-4 fully autonomous. 2

[19] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022. 1, 2

[20] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. 2, 3

[21] Guillaume Sanchez, Honglu Fan, Alexander Spangher, Elad Levi, Pawan Sasanka Ammanamanchi, and Stella Biderman. Stay on topic with classifier-free guidance. *arXiv preprint arXiv:2306.17806*, 2023. 3, 5, 8, 12

[22] Simon Kornblith, Lala Li, Zirui Wang, and Thao Nguyen. Guiding image captioning models toward more specific captions. In *ICCV*, pages 15259–15269, 2023. 2, 7, 10

[23] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. *arXiv preprint arXiv:2310.03744*, 2023. 2, 5, 7, 9, 10, 15

[24] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, Maxwell Nye, Augustus Odena, Arushi Somani, and Sağnak Taşırlar. Introducing our multimodal models, 2023. 2

[25] 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 *ECCV*, pages 740–755. Springer, 2014. 2, 7

[26] 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? *arXiv preprint arXiv:2307.06281*, 2023. 2, 6, 7, 10

[27] 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. 2, 6, 7, 10

[28] 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. 2

[29] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[30] 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 *ICML*, pages 8748–8763. PMLR, 2021. 2, 7, 10

[31] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv preprint arXiv:2306.14824*, 2023. 2

[32] Pan Zhang, Xiaoyi Dong, Bin Wang, Yuhang Cao, Chao Xu, Linke Ouyang, Zhiyuan Zhao, Shuangrui Ding, Songyang Zhang, Haodong Duan, Wenwei Zhang, Hang Yan, Xinyue Zhang, Wei Li, Jingwen Li, Kai Chen, Conghui He, Xingcheng Zhang, Yu Qiao, Dahua Lin, and Jiaqi Wang. Internlm-xcomposer: A vision-language large model for advanced text-image comprehension and composition, 2023. 2, 9

[33] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Michal Podstawski, Hubert Niewiadomski, Piotr Nyczyk, et al. Graph of thoughts: Solving elaborate problems with large language models. *arXiv preprint arXiv:2308.09687*, 2023. 2

[34] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, Xu Zhao, Min-Yen Kan, Junxian He, and Qizhe Xie. Decomposition enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*, 2023. 2

[35] Wei-Ge Chen, Irina Spiridonova, Jianwei Yang, Jianfeng Gao, and Chunyuan Li. Llava-interactive: An all-in-one demo for image chat, segmentation, generation and editing, 2023. 2

[36] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. *arXiv preprint arXiv:2308.00692*, 2023. 2

[37] Zeyi Sun, Ye Fang, Tong Wu, Pan Zhang, Yuhang Zang, Shu Kong, Yuanjun Xiong, Dahua Lin, and Jiaqi Wang. Alpha-clip: A clip model focusing on wherever you want, 2023. 2

[38] Songwei Ge, Taesung Park, Jun-Yan Zhu, and Jia-Bin Huang. Expressive text-to-image generation with rich text. In *ICCV*, 2023. 3

[39] Hila Chefer, Yuval Alaluf, Yael Vinker, Lior Wolf, and Daniel Cohen-Or. Attend-and-excite: Attention-based semantic guidance for text-to-image diffusion models. *arXiv preprint arXiv:2301.13826*, 2023. 3

[40] David Wan, Jaemin Cho, Elias Stengel-Eskin, and Mohit Bansal. Contrastive region guidance: Improving grounding in vision-language models without training. 2024. 3

[41] Linxi Zhao, Yihe Deng, Weitong Zhang, and Quanquan Gu. Mitigating object hallucination in large vision-language models via classifier-free guidance, 2024. 3

[42] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *NeurIPS*, 34:8780–8794, 2021. 3

[43] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv*, 2023. 4

[44] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, Wesam Manassra, Prafulla Dhariwal, Casey Chu, Yunxin Jiao, and Aditya Ramesh. Improving image generation with better captions. Technical report, OpenAI, 2023. 6, 7, 10, 16

[45] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. 6

[46] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023. 7

[47] Noam Rotstein, David Bensaid, Shaked Brody, Roy Ganz, and Ron Kimmel. Fusecap: Leveraging large language models to fuse visual data into enriched image captions. *arXiv preprint arXiv:2305.17718*, 2023. 7

[48] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, and Yejin Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021. 6, 10

[49] Yuechen Zhang, Jinbo Xing, Eric Lo, and Jiaya Jia. Real-world image variation by aligning diffusion inversion chain. In *NeurIPS*, 2023. 8

[50] Haozhe Zhao, Zefan Cai, Shuzheng Si, Xiaojian Ma, Kaikai An, Liang Chen, Zixuan Liu, Sheng Wang, Wenjuan Han, and Baobao Chang. Mmicl: Empowering vision-language model with multi-modal in-context learning, 2023. 10

[51] Ziyi Lin, Chris Liu, Renrui Zhang, Peng Gao, Longtian Qiu, Han Xiao, Han Qiu, Chen Lin, Wenqi Shao, Keqin Chen, Jiaming Han, Siyuan Huang, Yichi Zhang, Xuming He, Hongsheng Li, and Yu Qiao. Sphinx: The joint mixing of weights, tasks, and visual embeddings for multi-modal large language models. `https://github.com/Alpha-VLLM/LLaMA2-Accessory/blob/main/SPHINX/SPHINX_paper.pdf`, 2023. 10

[52] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 10