Appendix: ControlLLM: Augment Language Models with Tools by Searching on Graphs

Zhaoyang Liu[1,2], Zeqiang Lai[2], Zhangwei Gao[2,3], Erfei Cui[2,3], Ziheng Li[4], Xizhou Zhu[4,2], Lewei Lu[5], Qifeng Chen[1]✉, Yu Qiao[2], Jifeng Dai[4,6], and Wenhai Wang[7,2]✉

[1]HKUST [2]OpenGVLab, Shanghai AI Laboratory
[3]Shanghai Jiao Tong University [4]Tsinghua University [5]SenseTime
[6]BNRist [7]The Chinese University of Hong Kong

In the appendix, we present additional experiments, both objective and subjective, to validate the efficacy of ControlLLM. We also include visualizations of the distribution and word cloud for our constructed benchmark. Furthermore, we elaborate on the implementation details to better understand our approach.

1 Feature Comparisons

Table A1 presents a comprehensive feature comparison among various methods [8,9,13,14], highlighting ControlLLM's distinct advantages in the landscape of multi-modal interaction. Notably, "Multi-Solution" signifies the method's ability to provide multiple feasible solutions, granting users more options. "Pointing Device" signifies support for pointing devices such as the mouse, to enhance user experience. "Resource Type Awareness" indicates the method's capability to discern the type of resource in the context, ensuring more context-aware responses. In summary, ControlLLM emerges as the standout choice, excelling in various features. It offers a comprehensive set of tools in the domains of image, video, and audio. Moreover, its support for resource type awareness, multiple solutions, and pointing inputs demonstrates its adaptability and scalability, making it the highly versatile framework for diverse multi-modal interaction scenarios.

2 Additional Experiments

2.1 The efficacy of ToG.

As shown in Table 3, the greedy search is essentially a retrieval-based method that selects the best-matched tool assessed by LLMs. We can observe that there is a large performance gap between greedy search and adaptive search. This result directly proves the efficacy of our proposed ToG.

Table A1: Comparisons of features between different methods. The table shows that our framework supports more features that facilitate the user experience of multi-modal interaction. It proves the high scalability of our framework.

Features	**ControlLLM** (our work)	**HuggingGPT** [9]	**Visual ChatGPT** [13]	**InternGPT** [8]	**GPT4Tools** [14]
Image Perception	✓	✓	✓	✓	✓
Image Editing	✓	✓	✓	✓	✓
Image Generation	✓	✓	✓	✓	✓
Video Perception	✓	✓	✗	✓	✗
Video Editing	✓	✓	✗	✓	✗
Video Generation	✓	✓	✗	✓	✗
Audio Perception	✓	✓	✗	✗	✗
Audio Generation	✓	✓	✗	✗	✗
Multi-Solution	✓	✗	✗	✗	✗
Pointing Device	✓	✗	✗	✓	✗
Resource Type Awareness	✓	✗	✗	✗	✗

2.2 The performance of downstream tasks.

We conduct experiments mainly on two downstream tasks, *i.e.*, VQA in Table A2 and text-to-image generation in Table A3, to study the impacts on tool's performance, despite the fact that they are not jointly optimized with our framework. For VQA, we validate the performance on two widely used benchmarks: MMbench [7] and LLaVA^W: LLaVA-Bench (In-the-Wild) [6]. Since LLMs summarize and polish the response from the tool before replying to the user, we find ControlLLM can slightly improve the performance on LLaVA^W. However, we also notice there is a gap on MMBench. Given that this benchmark consists entirely of single-choice questions, ControlLLM will add extra words to explain the reason for its choice, which hinders the evaluation script from extracting the selected option. When directly using the output of the tool (see results in the middle of the table), the performance of ControlLLM is on par with the results from the original paper. In the task of text-to-image generation, ControlLLM can refine and enrich the text prompt for generation. Our method can achieve comparable or even higher performance than PIXART-α. Generally, our approach does not significantly impair the performance of the underlying tools.

2.3 The performance on a large scale benchmark.

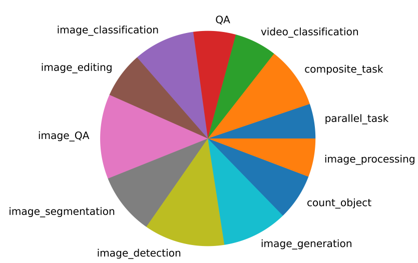
To further validate the efficacy of ControlLLM, we extend the benchmark used in main paper to a larger scale. The new large benchmark, *i.e.*, **Tool2K**, contains 2500 instructions across diverse domains. We here use **Tool100** to denote the benchmark used in the main paper for clarity. We showcase the distributions of benchmarks in Fig A1b and A1a as well as the word cloud of instructions in Fig A2. GPT-4 is employed to assess the solutions. Here is the evaluation protocol: (1) Tool Usage: Verify that the assistant uses the tools correctly to address the user's question. (2) Relevance: Ensure no irrelevant tools are included in the assistant's approach. (3) Arguments (tool inputs): Check that the arguments for the tools are correct and compatible in types and values. (4) Efficiency and

Table A2: The evaluation on visual question answering. * denotes the results are reproduced by ourselves. We utilize LLaVA [5] as our VQA tool in ControlLLM.

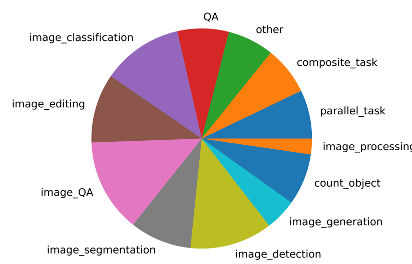
Methods	LLaVA^{W} [5] ↑	MMBench [7] ↑
LLaVA-1.5 [4]	63.4	64.3
LLaVA-1.5* [4]	68.6	64.1
ControlLLM	73.5	61.2

Table A3: The evaluation for text-to-image generation on T2I-CompBench [3]. As PIXART-α [5] is used as our text-to-image tool, we here only compare our ControlLLM with it.

Methods	Attribute Binding			Object Relationship		Complex ↑
	Color ↑	Shape ↑	Texture ↑	Spatial ↑	Non-Spatial ↑	
PIXART-α [1]	0.6886	0.5582	0.7044	0.2082	0.3179	0.4117
ControlLLM	0.7053	0.7796	0.7102	0.2121	0.3083	0.4220



(a) The of distribution of Tool100.



(b) The of distribution of Tool2k.

Fig. A1: The distribution of our benchmarks. Composite task represents the task that is composed of multiple task domains. The parallel task is using one instruction to handle several images or videos in a batch manner.

Effectiveness: Evaluate whether the solution provided by the assistant is efficient and effective. Based on this protocol, GPT-4 is required to give a score on the scale of 1-10 (1: not working/no sense; 10: perfect) for each solution. We compute the average score (termed **GPT-Score**) over the whole benchmark as our metric. Since HuggingGPT [9] can return a formatted solution and as well share a similar toolset in the names and usage of tools, we believe Table A4 can provide a fair comparison. As demonstrated, ControlLLM still achieves a superior GPT-Score in a larger benchmark.

In addition, we explore the impact of different LLMs on task decomposition and how it affects the task planning for ToG. We design a metric of decomposition accuracy (**DeAcc**) to assess the capability of task decomposition in the first stage of ControlLLM. As long as each field in sub-task is consistent with the

Table A4: Comparisons with HuggingGPT [9] on Tool2K.

Methods	GPT-Scores ↑
HuggingGPT [9]	5.24
GPT4Tools [14]	5.62
VisProg [2]	6.27
ViperGPT [10]	8.19
ControlLLM	8.68

Table A5: Impacts of LLMs on task decomposition. We conduct an ablation study of using different LLMs for task decomposition on Tool2K.

Methods	LLMs in Stage 1	DeAcc ↑	GPT-Scores ↑
ControlLLM	OPT-350M [15]	90.24%	7.11
	OPT-2.7B [15]	92.50%	8.51
	LLaMA-7B [12]	93.99%	8.68

Fig. A2: Word cloud of instructions over two benchmarks.

groundtruth, this decomposition result can be seen as correct. We can observe that the final GPT-Scores are strongly related to the capabilities of LLMs.

2.4 User study.

In order to subjectively evaluate the performance of our ControlLLM in comparison to other state-of-the-art methods, we conducted a user study collecting approximately 200 dialogue samples from various domains. To ensure unbiased results, we masked the names of the methods during the study. Participants are asked to select one or more acceptable results for each instruction, with the option to choose "*None of them*" if all results are deemed unacceptable. The satisfaction rate is defined as the number of times selected divided by the total number of participants in the user study. The results are summarized in A3. We received feedback from a total of 67 individuals, and the results demonstrate that ControlLLM outperforms other approaches by a significant margin. We observe some failure cases of ControlLLM can be blamed on task decomposition, which wrongly parses the user input. In addition, ToG may generate sub-optimal or invalid solutions for some challenging tasks.

2.5 Case Studies

We provide more cases across different modalities to validate the user experience of ControlLLM in practice. In Fig. A5, we present some cases of image perception, which involves analyzing and understanding the content of an image, such as detecting objects, counting objects, finding objects, segmenting objects, answering questions about the image, *etc*. These tasks require the system to invoke tools to process visual information and extract relevant features and labels from the image. Fig. A6 gives examples of image processing and editing, which assist users in processing or editing the image according to the instruction. Fig. A7 mainly focuses on image question answering and image generation, showing the graphic dialogue capability. In Fig. A8, we provide some multi-modal interaction cases on image, video, and audio domains. In addition, we also illustrate the capabilities of complicated scenarios with solutions searched by ToG during task planning in Fig. A9 and Fig. A10. These complex tasks involve combining multiple tools to find an advanced and creative solution path that can solve more challenging problems. It requires a system that can integrate different types of information and outputs from tools and generate comprehensive and meaningful responses based on execution results. These figures demonstrate the strong capabilities of ControlLLM in task planning for both simple and complicated scenarios. It thus leads to a better user experience.

3 ControlLLM-ChatGPT

In this variant, we implement language model $\mathcal{M}$ with ChatGPT. As such, we elaborately design a series of prompts for each module.

3.1 Task Decomposition

The prompt in Table A6 is designed for task decomposition in ControlLLM-ChatGPT. It guides the ChatGPT to decompose the user request into several subtasks. Table 1 in the main paper shows the output protocol.

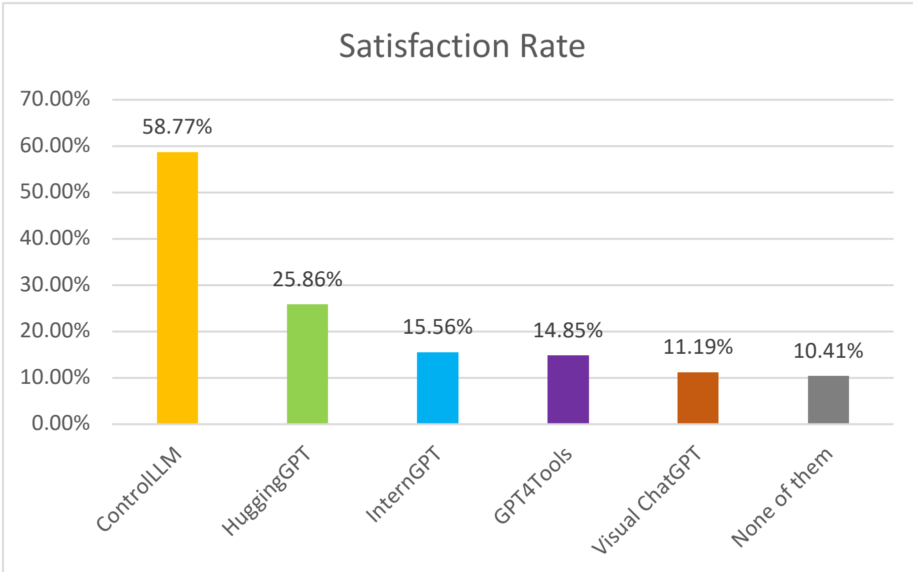


Fig. A3: User study.

3.2 Tool Assessment

Table A7 outlines a prompt design for the task of tool assessment, where the AI assistant evaluates tools' suitability for a given task. The output is captured in the JSON format, including reasons and a score. The scoring criteria range from 1 to 5, reflecting the tool's relevance to the task. The prompt emphasizes the connection between tool descriptions and task requirements. This prompt guides AI in making informed decisions when assessing tools' utility for a specific task.

3.3 Solution Expert

In this section, we delve into the core concept of the solution expert that streamlines the process of evaluating and selecting optimal solutions from all possible candidates. By systematically converting each solution into a formatted string description, the solution expert enables us to make informed decisions based on evaluated scores.

Solution Description Formatting. To facilitate the solution expert to comprehend the solution, we need to generate the description for each solution candidate. This involves transforming raw solution data into structured, formatted string descriptions. These descriptions encapsulate the information including functionality, inputs and output.

Solution Evaluation. The solution expert capitalizes on prompt engineering techniques to assess each solution based on subtask descriptions and formatted solution descriptions. The designed prompts guide language model $\mathcal{M}$

Table A6: The prompt for task decomposition. It is inspired by [9].

The following is a friendly conversation between a human and an AI. The AI is professional and parses user input to several tasks with lots of specific details from its context. If the AI does not know the answer to a question, it truthfully says it does not know. The AI assistant can parse user input to several tasks with JSON format as follows: <Solution>[“description”: task_description, “task”: [task_domain_1, task_domain_2], “id”: task_id, “dep”: dependency_task_id, “args”: [“type”: “text”, “image” or audio, “value”: text, image_url or <GEN>-dep_id], “returns”:[“type”: “segmentation”, “value”: “<GEN>-task_id”]]</Solution>. The “description” should describe the task in detail, and AI assistant can add some details to improve the user’s request without changing the user’s original intention. The special tag “<GEN>-dep_id” refers to the one generated text/image/audio/video/segmentation/mask in the dependency task (Please consider whether the dependency task generates resources of this type.) and “dep_id” must be in “dep” list. The special tag “<GEN>-task_id” refers to the one generated text/image/audio/video/segmentation/mask in this task and “task_id” should be in line with field “id” of this task. The “dep” field denotes the ids of the previous prerequisite tasks, which generate a new resource that the current task relies on. The “args” field and the “returns” field denote the input resources and output resources of this task, respectively. The type of resource must be in [“text”, “image”, “line”, “normal”, “hed”, “scribble”, “pose”, “edge”, “bbox”, “category”, “segmentation”, “audio”, “video”, “segmentation”, “mask”], nothing else. The “task” MUST be selected from the following options: “question-answering”, “visual-question-answering”, “image-generation”, “image-editing”, “image-perception”, “image-processing”, “audio-perception”, “audio-generation”, “audio-editing”, “video-question-answering”, “video-perception”, “video-generation”, “video-editing”, nothing else. Think step by step about all the tasks that can resolve the user’s request. Parse out as few tasks as possible while ensuring that the user request can be resolved. Pay attention to the dependencies and order among tasks. If some inputs of tools are not found, you cannot assume that they already exist. You can think of a new task to generate those args that do not exist or ask for the user’s help. If the user request can’t be parsed, you need to reply empty JSON []. You should always respond in the following format: <Solution><YOUR_SOLUTION></Solution> <YOUR_SOLUTION>should be strict with JSON format described above.

to evaluate the feasibility of each solution against the objective of the subtask. Through this process, we can assign scores to solutions, gauging their effectiveness and relevance to the task. It must ensure that the evaluation process is focused, targeted, and aligned with the subtask. The prompt template is shown in the Table A8.

Solution Ranking. The final aim of this module is to select the top-performing solutions. The optimal solution is identified as the highest score assessed in the solution evaluation. Given that sometimes the selected optimal solution may not meet the user requirements, we also provide several alternative solutions by setting a threshold score of 3. These solutions, which exhibit a

Table A7: The prompt for tool assessment.

Given a task and a tool, the AI assistant helps the system decide whether this tool can process the task. The assistant should focus more on the description of the model and give a score to each tool. The AI assistant respond with JSON format as follows: <Solution>"Thought": thought, "Score": score </Solution>. The "Thought" field records the model's thinking process step by step within 80 words, which gives the reasons why giving this score. The "Score" field denotes a score that assesses whether this tool is useful for this task. Score is in [1, 2, 3, 4, 5]. Here are the scoring criteria: "Score"=1: The tool is totally not related to the task and does not provide any useful output for solving the task. "Score"=2: The tool is somewhat not related to the task and may not provide any useful output for solving the task. "Score"=3: The tool is probably related to the task and provides some intermediate output that is partially helpful for solving the task, but it may not be the optimal one. "Score">3: The tool is closely or directly related to the task and provides an output that is mostly helpful for solving the task or that matches the returns of the task with regard to the type. In a nutshell, for the given task, the higher the score, the more useful the tool is. You should always respond in the following format: <Solution>SOLUTION </Solution>. \n'SOLUTION' should strictly comply with the SON format described above. Task description: "{{task}}".\n\n Here is the description of the tool "{{tool_name}}": \n{{tool_name}}: {{tool_description}}\nArgs: \n{{arguments}}\nReturns: \n{{returns}}\n\nThe above information may be useful for AI to make decision. Please refer to the scoring criteria and score the tool {{tool_name}} for this task. Notice that If the tool description contains keywords from the task description, the score of this tool should be greater than or equal to 3.

higher degree of alignment with the subtask's requirements, emerge as the most promising candidates for user preference.

Through collaborative efforts, the optimal solution expert ensures that solutions are appropriately tailored, optimized, and well-adapted to the task.

3.4 Resource Expert

In the algorithm of ToG, we encounter a challenge stemming from the potential presence of multiple instances of the same resource type within the available resource list. This challenge introduces complexity, making it difficult to straightforwardly deduce certain arguments for tools using pre-defined rules. As a result, we design a solution expert.

This module transforms the task of argument assignment into a fill-in-the-blank exercise. To achieve this, we design a resource expert crafts with prompts that not only incorporate the task description but also include the available resource list. In this manner, a language model $\mathcal{M}$ is employed to dynamically complete the missing parameters within a solution by interacting with the contextual information presented. We put the prompt template in the Table A9.

Table A8: The prompt for solution expert.

Given a task and a solution, The AI assistant needs to score the solution and respond in JSON format. Please notice that the AI assistant should think. The AI assistant should pay more attention to the relevance between the description of each tool in the solution and task. The AI assistant respond with JSON format as follows: <Solution>{"Thought": "thought", "Score": score}</Solution>. "Thought" field records the model's thinking process step by step within 80 words, which gives the reasons why giving this score. "Score" field denotes a score that assesses whether this tool is helpful for this task. "Score" is in [1, 2, 3, 4, 5]. Here are the scoring criteria: "Score"=1: The solution is totally not related to the user's request and can not solve the task. "Score"=2: The solution is somewhat not related to the user's request and may not solve the task. "Score"=3: The solution is probably related to the user's intention and may solve the task, but it may not be the optimal one. "Score">3: The solution is closely or directly related to what the user wants and could satisfactorily solve the task. In a nutshell, the higher the score, the greater the likelihood of the solution solving the given task. You should always respond in the following format: \n<Solution>'SOLUTION' </Solution>\n'SOLUTION' should strictly comply with the JSON format described above. \nUser's request: "{{request}}" Task description: "{{task}}". Here is the description of the solution: {{solution}} Please refer to the scoring criteria and score this solution based on the task description. You should think carefully before scoring the solution. Notice that If the keywords in the solution are close in meaning to the keywords in the task description, then the score of this solution is at least 3.

Table A9: The prompt for resource expert.

Some tools have missing input arguments, and the AI assistant needs to infer these missing inputs from the context. Please notice that the AI assistant should never fake the resources that do not exist. The returned input arguments should be JSON format as follows: [{"image": "xxx.png"}, {"bbox": "<GEN>-detr-bbox-0"}, "text": "<text>"]. AI assistant should always respond in the following format: \n"<Explanation>[briefly explain your choice here]</Explanation><Solution>'SOLUTION' </Solution>". \n'SOLUTION' should be strictly in the JSON format described above. \nUser's request: \n"{{request}}"\nTask: \n"{{task_description}}". \n<Resources>: \n{{resources}}. \nWe use {{tool_name}} to solve this task: \n'{{tool_name}}': {{tool_description}} \nArgs: {{arguments}} \nReturns: {{returns}} \nFor the type of "text", the AI assistant should summarize the content from the context based on the task and the tool's description. For other types of input, the AI assistant needs to select the inputs from <Resources>. Now we prepare the inputs for {{tool_name}}: {{input}}. Please complete these inputs and return the completed inputs with the format described above like: <Solution>'SOLUTION' </Solution>.

4 ControlLLM-LLaMA

For ControlLLM-LLaMA, we use the LLaMA-7b [12] as language model $\mathcal{M}$ to solve the problems in task decomposition, tool assessment, solution expert, resource expert.

Table A10: The Prompt Design in Response Generation. We here refer to the prompts from [9] to generate a user-friendly response.

Your name is ControlLLM, an AI-powered assistant. For the user's request, the system executes the solution and collects the results based on the following workflow. You need to respond to user requests based on the following information. Here is the information for your reference. ## User Request {{request}} ## Workflow and Execution Results {{solution}} ## Summarized Results {{results}} You must first answer the user's request in a straightforward manner. Some of the results may not be accurate and need you to use your judgment in making decisions. Then please explain your workflow, including the tools and returned results for the request, in a friendly way. If the answers contain file paths, you have to repeat the complete file path. Only if there is nothing in the Summarized Results, you need to tell the user you can not finish the task.

4.1 Instruction Generation

The first step to train $\mathcal{M}$ is to construct the instruction corpus. We here opt for ChatGPT-3.5 to generate the training corpus. The following steps will elaborate on the details of instructions generation for task decomposition, tool assessment, solution expert, and resource expert, respectively.

For task decomposition, we generate two different types of instructions as follows: 1) Basic instructions, where they only contain one subtask after task decomposition. We set some seed instructions with ground-truth results of task decomposition, which serve as initial templates for generating more diverse instructions. Then, we use ChatGPT to generate more diverse instructions based on the pre-defined seed instructions. During the generation process, we center on the seed instructions and produce more instructions using more diverse expressions and styles. These instructions need to share the task decomposition results with the seed instructions as ground truth. 2) Compound instructions, which involve multiple subtasks and intermediate resources. We simply assemble the basic instructions into the compound instructions in a coherent and logical manner. It aims to enhance the improve the complex interaction capability of the system by enabling the model to handle user requests that span multiple domains and require multiple steps of processing. We here generate almost 100k instructions for training. The instructions generated in this step will be used in the following tasks as well.

For the tool assessment, solution expert, and resource expert, we use prompts in Table A7, A8 and A9 to collect the output from ChatGPT by running ControlLLM on the instructions generated above. Unlike directly generating the solution, these tasks only involve making a decision, like scoring the tools or solutions based on the input, so they are relatively simple, and ChatGPT, with strong zero-shot capabilities, can easily solve them. Therefore, we opt to directly distill the knowledge of ChatGPT by using prompt techniques. Through the experiments, we verify the feasibility of this strategy.

4.2 Training Recipes

We follow the training protocol in [11], where LLaMA [12] is used as an alternative choice for our language model $\mathcal{M}$. It is finetuned for three epochs with the initial learning rate 2e-5 and consine decay. We fix the training batch size as 128 by adaptively setting "gradient_accumulation_steps". The whole training procedure is on 8xA100 GPUs.

5 ControlLLM-Mix

In practice, we find ControlLLM-ChatGPT has difficulty in task decomposition, especially for hard instructions. In addition, ControlLLM-LLaMA is good at task decomposition due to an instruction-tuned language model $\mathcal{M}$ while other modules are slightly inferior to ChatGPT. Because we finetune $\mathcal{M}$ by distilling the knowledge from ChatGPT to assess tools, ranking solutions, and assign arguments. As a result, we design ControlLLM-Mix to integrate the benefits from the other variants. For task decomposition, we use the same method from ControlLLM-LLaMA to finetune LLaMA-7B to decompose user requests. For the remaining modules, we directly utilize the ChatGPT, sharing the same prompt design from ControlLLM-ChatGPT. In experiments, ControlLLM-Mix achieves the best performance.

6 Response Generation

We design a prompt template for the Response Generation in Table A10. In this task, the AI assistant is tasked with explaining the process and outcomes using input and inference results. The AI is instructed to respond directly to the user's request, followed by describing the task's procedure, offering analysis, and presenting model inference results using a first-person perspective. If the results involve file paths, the complete path should be provided, or if there are no results, the AI should communicate its inability. The prompt sets the context for generating informative and user-understandable responses during response generation.

7 Resource Types and Tools

We initially define a type set containing a series of resource type descriptors, such as "text", "tags", "html", "image", "video", "audio", "segmentation", "edge", "line", "hed", "canny", "scribble", "pose", "depth", "normal", "mask", "point", "bbox" and

Table A11: A subset of domains and tools in our tool kit.

Domains	Tools
question-answering	question_answering, image_question_answering
NLP	summarization, title_generation, text_to_tags, text_to_text_generation, sentiment_analysis
image-perception	object_detection, image_captioning, visual_grounding, image_classification, segment_anything, instance_segmentation, segment_by_points
image-generation	text_to_image, image_to_image, line_text_to_image, hed_text_to_image, scribble_text_to_image, pose_text_to_image, segmentation_text_to_image, edge_text_to_image, depth_text_to_image, normal_text_to_image
image-editing	text_image_editing, image_inpainting, image_cropping, mask_image, highlight_object_on_image
image-processing	image_to_edge, image_to_line, image_to_hed, image_to_scribble, image_to_pose, image_to_depth, image_to_normal
video-perception	video_classification, video_captioning
video-processing	dub_video, video_to_webpage
video-generation	image_audio_to_video, image_to_video, text_to_video
audio-perception	audio_classification
audio-generation	text_to_music, text_to_speech, audio_to_Audio

"category". The type set is easy to extend depending on the toolkit. The types of inputs of tools must fall into the pre-defined type set.

In Table A11, we exhibit lots of tools supported by our framework across different domains, including natural language, image, audio, video, *etc*. The whole system will continue to evolve. As depicted in Fig. A4, we visualize the topological structure of tool graph and showcase how it looks like.

8 Metrics for Tool Selection

A) Irrelevant Tool Inclusion Rate (*abbr. IR*):

$$F(s^p) = \begin{cases} \text{true}, & s^p \text{ contains the irrelevant tools} \\ \text{false}, & \text{otherwise} \end{cases}, \tag{1}$$

$$IR = \frac{\sum_i^{|S^p|} \mathbb{I}(F(S_i^p))}{|S^p|}, \tag{2}$$

where $\mathbb{I}$ is indicator function, $|\cdot|$ represents the number of elements in a set, and S^p denotes all predicted solutions on our benchmark. It measures the proportion of the predicted solutions that contain the irrelevant tools. A higher

Table A12: Test samples of instruction in the benchmark.

Easy
1. Please determine if the image_1.png contains a platyhelminth?
2. How can I design a sleep monitoring system in C# that can accurately detect a baby's specific sleep stages and predict when they will enter a light sleep stage within the next hour? And once this prediction is made, how can I alert the parent or caregiver that the baby will be waking up soon and suggest soothing methods to ease the transition from sleep to wakefulness? Also, how can I modify the statement "The baby is sleeping" to reflect this predictive system in C#?
3. Please extract the scribble result for the image in image_2.png"
4. With the HED result image_3.png, generate a new image that features a zoo landscape with various animals, a couple with their children, and a fountain.
5. Given the image image_4.png, What is unique about the window in the room?
Medium
6. Can you generate a new image that has a similar layout to the file named 'image_5.png'? I'm particularly interested in the positioning of the elements. The new image should have the same arrangement of elements and their positioning.
7. Generate a new image conditioned on the segmentation from image_6.png and the new image contains a majestic mountain range with a clear blue sky and a herd of wild horses running free.
8. Take away the umbrella from the picture image_7.png.
9. Crop out the baseball glove in image_8.png
10. Provide me with a mask that separates the bear from the rest of the image_9.png?
Hard
11. provide the number of umbrellas presented in the image_10.png, image_11.png, image_12.png, image_13.png, image_14.png
12. Can you elaborate on the elements of the image_15.png, image_16.png and image_17.png provided?
13. Erase the laptop from the image_18.png,image_19.png and image_20.png
14. Create a new image using the segmentation from image_21.png that showcases a cozy cabin in the woods with a dog and a cat, surrounded by snow-covered trees. Can you crop out the dog from given image? I'm looking for the cat in the image file, can you guide me to it?
15. Can you determine whether image_22.png contains a mouse? Please provide a list of all the objects present in the image, with a special emphasis on the killer. Is image_23.png displaying a banana? As for the image, what skills are important for improving one's performance in the depicted scenario?

IR indicates that the method tends to include more unnecessary tools, potentially hindering effective task planning. This metric gauges the performance of methods by excluding irrelevant tools.

B) Necessary Tool Inclusion Rate (*abbr.* NR):

$$H(s^p) = \begin{cases} \text{true}, & \text{Solution } s^p \text{ contains necessary tools} \\ \text{false}, & \text{otherwise} \end{cases}, \tag{3}$$

$$NR = \frac{\sum_i^{|S^p|} \mathbb{I}(H(S_i^p))}{|S^p|}. \tag{4}$$

The necessary tools play a critical role in solving the user request. For example, if users want to know the position of a specific object, the object detection tool is necessary. This metric measures the proportion of solutions that contain the necessary tools for solving the task but without considering whether the arguments of tools are correct. It checks whether the solution has all the necessary tools that can produce the expected output. A high NR value means that the method has a strong ability in task planning and finding the right tools for the user's request.

8.1 Metrics for Argument Assignment

A) Resource Hallucination Rate (*abbr.* HR):

$$P(s^p) = \begin{cases} \text{true}, & s^p \text{contains false resources} \\ \text{false}, & \text{otherwise} \end{cases}, \tag{5}$$

$$HR = \frac{\sum_i^{|S^p|} \mathbb{I}(P(S_i^p))}{|S^p|}. \tag{6}$$

This indicator reveals the extent of hallucination when inferring the arguments for tools. It checks whether all the arguments of the tools exist in the input resources or not. A low HR value means that the method rarely leads to hallucinations that are common in LLMs.

B) Resource Type Consistency Rate (*abbr.* CR):

$$Q(s^p) = \begin{cases} \text{true}, & \text{No resource type conflict in } s^p \\ \text{false}, & \text{otherwise} \end{cases}, \tag{7}$$

$$CR = \frac{\sum_i^{|S^p|} \mathbb{I}(Q(S_i^p))}{|S^p|}. \tag{8}$$

This metric examines whether the types of resources used as inputs for the predicted solution match those of the corresponding tools. It evaluates the method's ability to ensure consistency between argument types and tools. A high CR value means that the method can correctly infer and assign arguments for each tool.

8.2 Solution Evaluation

The Solution Evaluation (*abbr. SE*):

$$W(s^p) = \begin{cases} \text{true}, & s^p \text{ can solve the task} \\ \text{false}, & \text{otherwise} \end{cases}, \tag{9}$$

$$SE = \frac{\sum_i^{|S^p|} \mathbb{I}(W(S_i^p))}{|S^p|}. \tag{10}$$

This metric measures the success rate of all generated solutions on our benchmark, regardless of whether it contains irrelevant tools, as long as the chain of tool invoking outputs the information that is able to solve the task. A higher score in the solution evaluation indicates that the method is able to provide an effective solution to user requests.

In summary, these intuitive metrics together provide a comprehensive assessment of tool-augmented LLMs in terms of tool selection, argument assignment, and overall effectiveness in addressing user queries.

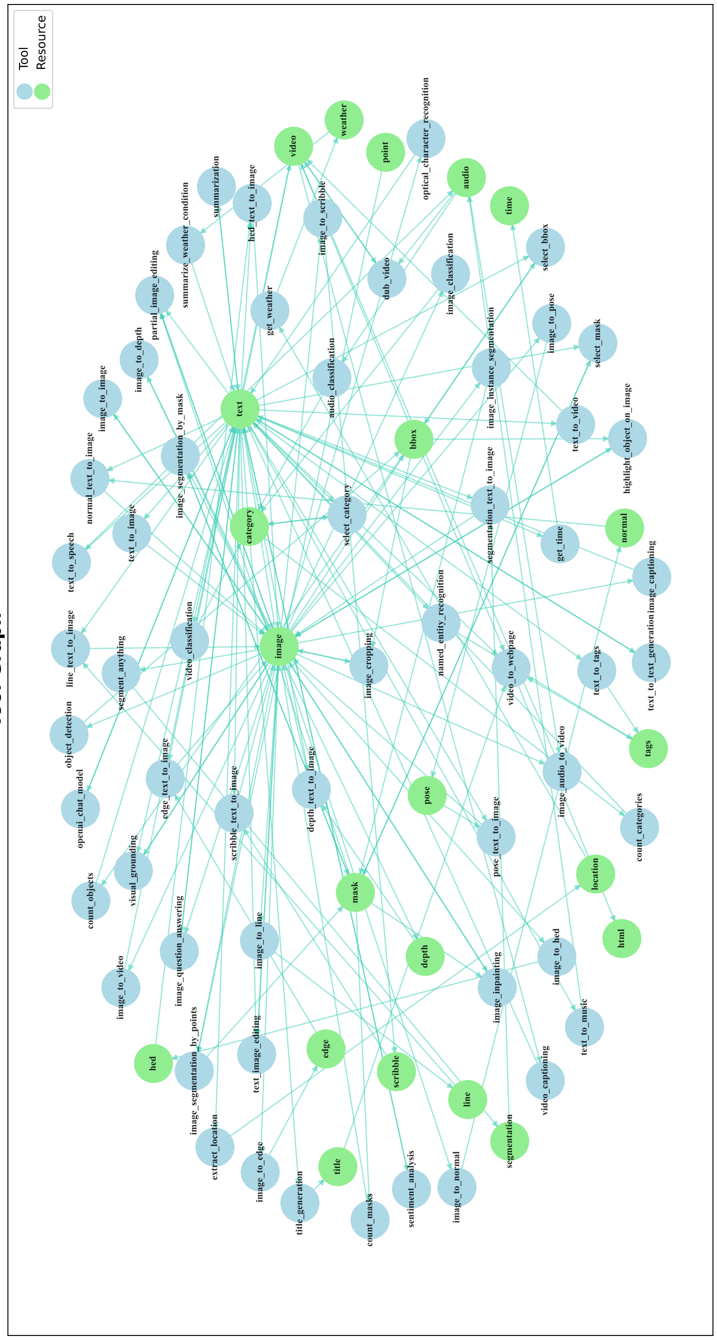


Fig. A4: The visualization of tool graph.

image-perception

User: Provide me with the count of bears in the input image

AI: The count of bears in the input image is 3.
Here is a summary of the workflow and results for your request:
1. The image was processed using an ***object detection*** tool, which detected the presence of bears in the image.
2. The bounding boxes around the detected bears were selected by ***bounding boxes selection*** tool.
3. The count of bears was then determined using a ***count objects*** tool.
4. The result of the count objects tool indicated that there are 3 bears in the image.

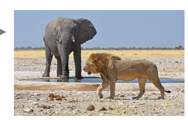
User: I'm trying to find the umbrella in the image, can you assist me?

AI: Yes, I can assist you in finding the umbrella in the image. After analyzing the image, I have identified the location of the umbrella and cropped it for better visibility.
To achieve this, I used the following tools in my workflow:

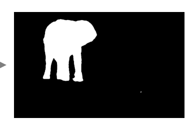
1. ***Object Detection***: This tool was used to detect various objects in the image, including the umbrella. The tool provided a list of detected objects with their corresponding labels and bounding box coordinates.
2. ***Select Bounding Box***: This tool was used to select the bounding box corresponding to the umbrella from the list of detected objects.
3. ***Image Cropping***: This tool was used to crop the image based on the selected bounding box of the umbrella. The result is a new image that focuses specifically on the umbrella.

I hope this explanation clarifies the steps I took to find and extract the umbrella from the image. If you have any further questions, feel free to ask!

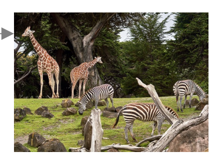
User: Give me the mask of elephant in the image.



AI: I have generated a mask of an elephant for you.
To generate this mask, I used an ***image instance segmentation*** tool on the image.
The tool identified different objects in the image and created masks for them. Then, I used a ***mask selection*** tool to specifically select the mask for the elephant based on the condition provided.
Please note that the result may not be completely accurate, as it relies on the performance of the tools used.



User: Can you highlight the giraffe for me in the image?



AI: I have highlighted the giraffe for you.

To fulfill your request, I followed the following workflow:
1.First, I used an ***object detection*** tool to locate the elephant in the image.
2.Then, I used a ***selection tool*** to choose the bounding box that contains the elephant.
3.Finally, I used an ***image highlighting*** tool to highlight the selected bounding box on the image.
The output of this workflow is the highlighted image, which shows the giraffe highlighted.

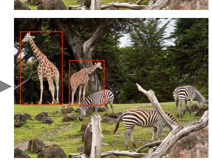


Fig. A5: Case study on different task domains (a).

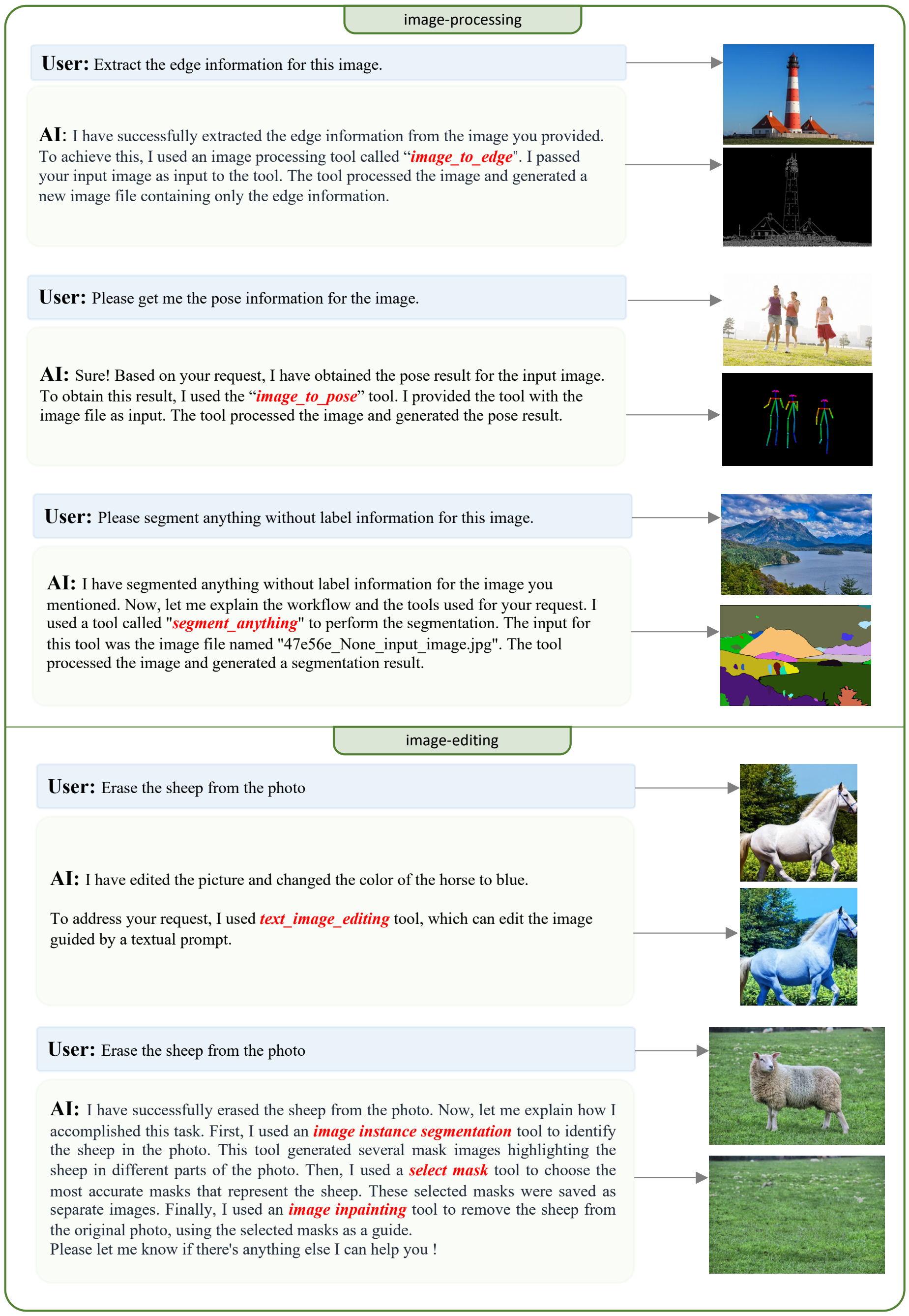


Fig. A6: Case study on different task domains (b).

image-question-answering

User: Why is this image so funny?

AI: The image is funny because it features two toy frogs, one of which is holding a camera and taking a picture of the other. The scene is set in a grassy field, and the toy frogs are posed in a way that mimics human behavior, specifically a photographer capturing a subject. The humor comes from the unexpected and anthropomorphic portrayal of the toy frogs, as it is not common to see inanimate objects, such as toys, engaging in activities like photography. This playful and whimsical scene can evoke amusement in the viewer.
I used an ***image question answering*** tool. The tool analyzed the image and determined why this image is so funny.

User: Given the image, is there going to be a potentially dangerous situation in the picture?

AI: There is a potentially dangerous situation as a man is riding on the back of an elephant, while another elephant is walking through the water. The presence of two elephants in close proximity to each other and the man could pose a risk of accidents or injuries, especially if the elephants become agitated or frightened. It is essential for the man to maintain control over the elephant he is riding and ensure that the other elephant is calm and well-behaved to minimize any potential dangers. To solve your question, I used an ***image question answering*** tool.

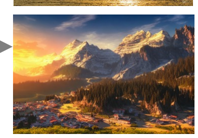
User: how many people can be seen riding the elephant in the image?

AI: In the image, there is one person riding on the back of the elephant. I used an ***image question answering*** tool to answer your question.

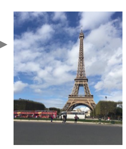
image-generation

User: I want a new image that has mountains and sunset. Can you use the depth result from input image to create it?

AI: The new image is generated. The workflow involved two tools: "***image_to_depth***" and "***depth_text_to_image***".
The first tool, "***image_to_depth***", used an input image to generate a depth result.
The second tool, "***depth_text_to_image***", took a text input of "The new image should have mountains and sunset." and a depth map (generated by the previous tool)



User: please generate a new image based the edge from this image. The new image shows a beautiful tower in the rain.



AI: The new image generated.
Workflow:
The image provided was processed using an ***image-to-edge*** tool, which generated an edge representation of the image.
The edge representation was then used along with the text "The new image shows a beautiful tower in the rain" in an ***edge-text-to-image*** tool. This tool combined the edge representation and the text to generate a new image.

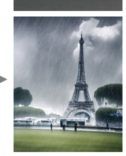


Fig. A7: Case study on different task domains (c).

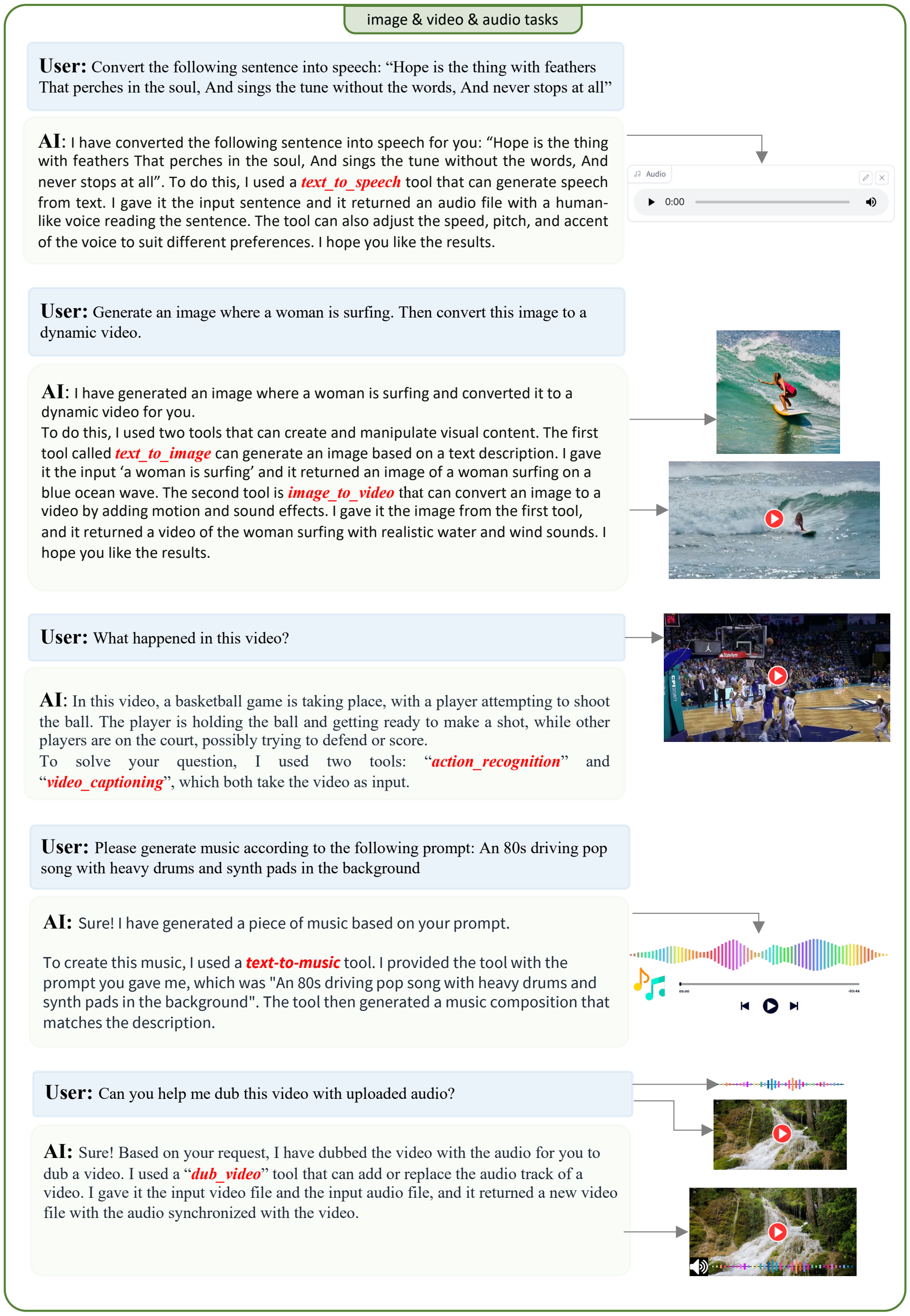


Fig. A8: Case study on different task domains (d).

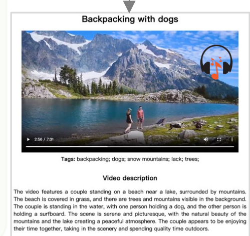
complex tasks

User: Please dub the video.mp4 with a suitable background music and then generate a HTML web page to introduce this video.

Task planning

[Action(tool_name='***video_captioning***', inputs={'video': 'video.mp4'}, outputs=['<TOOL-GEN>-text-0']), Action(tool_name='***text_to_music***', inputs={'text': '<TOOL-GEN>-text-0'}, outputs=['<TOOL-GEN>-text_to_music-audio-0']), Action(tool_name='***dub_video***', inputs={'video': "video.mp4 ", 'audio': "<TOOL-GEN>-text_to_music-audio-0",}, outputs=['<TOOL-GEN>-dub_video-video-0'])]
[Action(tool_name='***title_generation***', inputs={'text': '<TOOL-GEN>-text-0'}, outputs=['<TOOL-GEN>-text-1']), Action(tool_name='***text_to_tags***', inputs={'text': '<TOOL-GEN>-text-0'}, outputs=['<TOOL-GEN>-tags-0']), Action(tool_name='***video_to_webpage***', inputs={'video': '<TOOL-GEN>-dub_video-video-0', 'title': '<TOOL-GEN>-text-1', 'tags': '<TOOL-GEN>-tags-0', 'description': '<TOOL-GEN>-text-0'}, outputs=['<GEN>-0'])]

AI: Sure, I can generate a web page for your input video.
I have executed a workflow using several tools to generate the desired web page. The ***video_captioning*** tool was used to generate a text description for the input video. The ***dub_video*** tool combined the input video with the generated audio to create a dubbed video.
The ***title_generation*** tool was used to generate a title for the video based on the generated text.
The ***text_to_tags*** tool generated a set of tags related to the video content.
Finally, the ***video_to_webpage*** tool was used to generate a web page for the video, including the video itself, the generated title, tags, and description.
The resulting web page includes the following elements:
1) A title prompt asking for a title to be generated within 10 words. 2) A video player to play the input video. 3) Tags related to the video content. 4) A video description.



User: Generate a new image based on prompt. Prompt: "A serene and beautiful landscape with a calm lake reflecting the blue sky and white clouds. There are some green hills and trees on the shore". Then help me generate a promotional video for this generated image.

Task planning

[Action(tool_name='***text_to_image***', inputs={'text': 'A serene and beautiful landscape with a calm lake reflecting the blue sky and white clouds. There are some green hills and trees on the shore, and a small wooden boat floating on the water.'}, outputs=['<TOOL-GEN>-text_to_image-image-0']), Action(tool_name='***image_captioning***', inputs={'image': '<TOOL-GEN>-text_to_image-image-0'}, outputs=['<TOOL-GEN>-text-0']), Action(tool_name='***text_to_speech***', inputs={'text': '<TOOL-GEN>-text-0'}, outputs=['<TOOL-GEN>-text_to_speech-audio-0']), Action(tool_name='***image_audio_to_video***', inputs={'image': "<TOOL-GEN>-text_to_image-image-0", 'audio': "<TOOL-GEN>-text_to_speech-audio-0"}, outputs=['<GEN>-0'])]

AI: Sure! I can help you generate a video to introduce an image. Based on the information provided, I have used a combination of tools to fulfill your request.
First, I used the "***text_to_image***" tool to convert your description into an image. The tool created an image of a serene and beautiful landscape with a calm lake reflecting the blue sky and white clouds. There are also green hills and trees on the shore, as well as a small wooden boat floating on the water.
Next, I used the "***image_captioning***" tool to generate a textual description of the image. The tool analyzed the image and produced a caption that best describes it.
Then, I used the "***text_to_speech***" tool to convert the generated textual description into an audio file. This tool transformed the text into speech, allowing for a more engaging experience.
Finally, I used the "***image_audio_to_video***" tool to combine the generated image and audio into a video. The tool merged the image and audio together, creating a video that introduces the image.
The result of this workflow is a video file. This video showcases the image described in your initial request and includes the audio narration of its description.

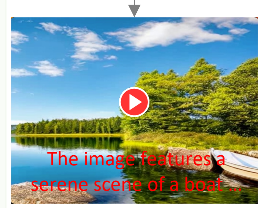
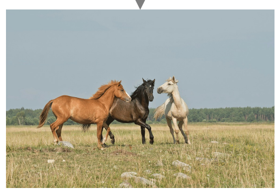
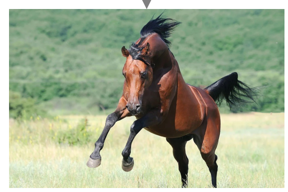


Fig. A9: Case study on different task domains (e).

complex tasks

User: Count horse in the image1.png, image2.png and image3.png





Task planning

[
Action(tool_name='***object_detection***', inputs={'image': 'image1.png'}, outputs=['<TOOL-GEN>-object_detection-bbox-0']), Action(tool_name='***select_bbox***', inputs={'bbox_list': '<TOOL-GEN>-object_detection-bbox-0', 'condition': 'bear'}, outputs=['<TOOL-GEN>-select_bbox-bbox-0']),
Action(tool_name='***count_objects***', inputs={'bbox_list': '<TOOL-GEN>-select_bbox-bbox-0'}, outputs=['<GEN>-0']),
],
[
Action(tool_name='***image_question_answering***', inputs={'text': 'Provide me with the count of bears in the input image', 'image': 'image1.png'}, outputs=['<GEN>-1'])
],
[
Action(tool_name='***object_detection***', inputs={'image': 'image2.png'}, outputs=['<TOOL-GEN>-object_detection-bbox-1']),
Action(tool_name='***select_bbox***', inputs={'bbox_list': '<TOOL-GEN>-object_detection-bbox-0', 'condition': 'bear'}, outputs=['<TOOL-GEN>-select_bbox-bbox-1']),
Action(tool_name='***count_objects***', inputs={'bbox_list': '<TOOL-GEN>-select_bbox-bbox-0'}, outputs=['<GEN>-2']),
],
[
Action(tool_name='***image_question_answering***', inputs={'text': 'Provide me with the count of bears in the input image', 'image': 'image2.png'}, outputs=['<GEN>-3'])
]
[
Action(tool_name='***object_detection***', inputs={'image': 'image3.png'}, outputs=['<TOOL-GEN>-object_detection-bbox-1']),
Action(tool_name='***select_bbox***', inputs={'bbox_list': '<TOOL-GEN>-object_detection-bbox-0', 'condition': 'bear'}, outputs=['<TOOL-GEN>-select_bbox-bbox-1']),
Action(tool_name='***count_objects***', inputs={'bbox_list': '<TOOL-GEN>-select_bbox-bbox-0'}, outputs=['<GEN>-4']),
],
[
Action(tool_name='***image_question_answering***', inputs={'text': 'Provide me with the count of bears in the input image', 'image': 'image3.png'}, outputs=['<GEN>-5'])
]

AI: Based on the information provided, the count of horses in the two images is eight.
To provide you with this information, I followed a workflow using various tools. First, I used an ***object detection*** tool to detect objects in the images. Then, I used a tool called "***select_bbox***" to select the bounding boxes that contain the desired object, which in this case is a horse. Next, I used a tool called "***count_objects***" to count the number of horses based on the selected bounding boxes. For both two images, I used the same solution to count the horses. The "***image_question_answering***" tool is also used to double check the result. The results of this workflow indicate that there are four horses in image1.png, three horses in image2.png and one horse in image3.png. So, there are eight horses in total. However, please note that these results may not be completely accurate, as they are generated using automated tools and may require human judgement for accuracy.

Fig. A10: Case study on different task domains (f).

References

1. Chen, J., Yu, J., Ge, C., Yao, L., Xie, E., Wu, Y., Wang, Z., Kwok, J., Luo, P., Lu, H., et al.: Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis. arXiv preprint arXiv:2310.00426 (2023)
2. Gupta, T., Kembhavi, A.: Visual programming: Compositional visual reasoning without training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14953–14962 (2023)
3. Huang, K., Sun, K., Xie, E., Li, Z., Liu, X.: T2i-compbench: A comprehensive benchmark for open-world compositional text-to-image generation. Advances in Neural Information Processing Systems **36** (2024)
4. Liu, H., Li, C., Li, Y., Lee, Y.J.: Improved baselines with visual instruction tuning. arXiv preprint arXiv:2310.03744 (2023)
5. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. In: NeurIPS (2023)
6. Liu, H., Li, C., Wu, Q., Lee, Y.J.: Visual instruction tuning. arXiv preprint arXiv:2304.08485 (2023)
7. Liu, Y., Duan, H., Zhang, Y., Li, B., Zhang, S., Zhao, W., Yuan, Y., Wang, J., He, C., Liu, Z., et al.: Mmbench: Is your multi-modal model an all-around player? arXiv preprint arXiv:2307.06281 (2023)
8. Liu, Z., He, Y., Wang, W., Wang, W., Wang, Y., Chen, S., Zhang, Q., Lai, Z., Yang, Y., Li, Q., Yu, J., et al.: Interngpt: Solving vision-centric tasks by interacting with chatbots beyond language. arXiv preprint arXiv:2305.05662 (2023)
9. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. arXiv preprint arXiv:2303.17580 (2023)
10. Surís, D., Menon, S., Vondrick, C.: Vipergpt: Visual inference via python execution for reasoning. Proceedings of IEEE International Conference on Computer Vision (ICCV) (2023)
11. Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B.: Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca` (2023)
12. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
13. Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671 (2023)
14. Yang, R., Song, L., Li, Y., Zhao, S., Ge, Y., Li, X., Shan, Y.: Gpt4tools: Teaching large language model to use tools via self-instruction (2023)
15. Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P.S., Sridhar, A., Wang, T., Zettlemoyer, L.: Opt: Open pre-trained transformer language models (2022)