CLOVA: A Closed-LOop Visual Assistant with Tool Usage and Update

Zhi Gao[1,2], Yuntao Du[2], Xintong Zhang[2,3], Xiaojian Ma[2],
Wenjuan Han[3], Song-Chun Zhu[1,2,4], Qing Li[2] ✉

[1]School of Intelligence Science and Technology, Peking University [2]State Key Laboratory of General Artificial Intelligence, BIGAI
[3]Beijing Jiaotong University [4]Department of Automation, Tsinghua University

https://clova-tool.github.io

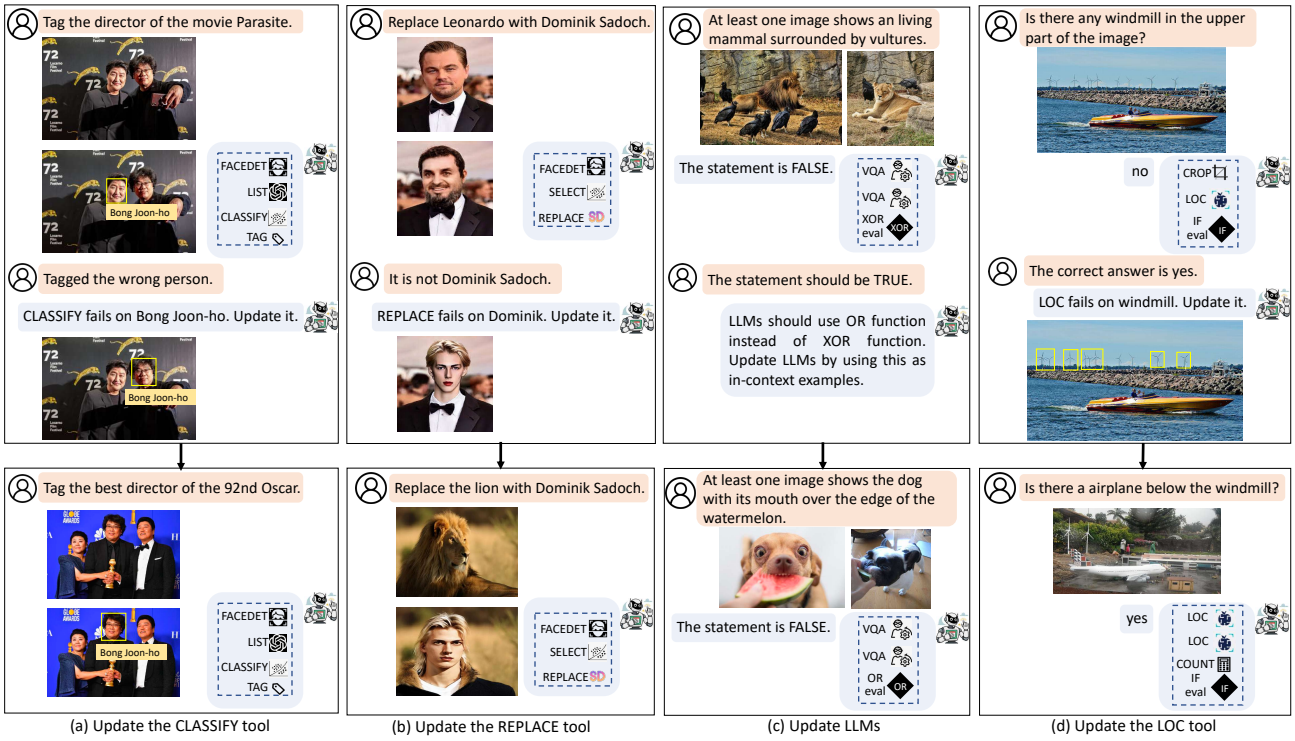

Figure 1. CLOVA is a general visual assistant that updates both LLMs and visual tools via inference, reflection, and learning in a closed-loop framework. During inference, CLOVA uses LLMs to integrate visual tools to accomplish given tasks. In reflection, CLOVA identifies tools that require updating based on human feedback. Finally, in learning, CLOVA collects data and updates the tools accordingly.

Abstract

Utilizing large language models (LLMs) to compose off-the-shelf visual tools represents a promising avenue of research for developing robust visual assistants capable of addressing diverse visual tasks. However, these methods often overlook the potential for ***continual learning****, typically by freezing the utilized tools, thus limiting their adaptation to environments requiring new knowledge. To tackle this challenge, we propose CLOVA, a Closed-LOop Visual Assistant, which operates within a framework encompassing inference, reflection, and learning phases. During the inference phase, LLMs generate programs and execute corresponding tools to complete assigned tasks. In the reflection phase, a multimodal global-local reflection scheme analyzes human feedback to determine which tools require updating. Lastly, the learning phase employs three flexible approaches to automatically gather training data and introduces a novel prompt tuning scheme to update the tools, allowing CLOVA to efficiently acquire new knowledge. Experimental findings demonstrate that CLOVA surpasses existing tool-usage methods by 5% in visual question answering and multiple-image reasoning, by 10% in knowledge tagging, and by 20% in image editing. These results underscore the significance of the continual learning capability in general visual assistants.*

✉ Corresponding author: Qing Li (dylan.liqing@gmail.com).

1. Introduction

The advancement of large language models (LLMs) [46, 67] alongside various visual tools (e.g., neural networks and OpenCV functions) [6, 27, 44, 55, 56] offers feasible avenues for constructing general visual assistants. When confronted with a task accompanied by language instructions, a common approach involves harnessing LLMs to generate programs, which are then executed using readily available visual tools to solve the task as dictated by the generated program [7, 10, 32, 40, 60, 65]. For instance, when posed with the query "*What is the person to the left of the umbrella doing?*", a viable solution entails LLMs sequentially performing the following steps: (1) utilizing a detection tool to locate the umbrella, (2) cropping the image region to the left of the umbrella, (3) using the detection tool to locate the person, and (4) finally querying a Visual Question Answering (VQA) tool with the question "*What is the person doing?*". Being highly compositional, such tool-usage methods demonstrate impressive performance and appealing explainability in tackling complex reasoning tasks, including VQA [11, 29, 30], mathematical reasoning [13, 14], and image editing [38, 72, 77, 78].

However, the potential for continual learning has been largely overlooked in existing tool-usage methods. Most of them simply freeze the used tools, which limits their applicability to environments where new knowledge is required, as depicted in Fig. 1. For instance, a user might instruct a visual assistant to label the face of the movie director Bong Joon-ho in a photograph. However, if the face recognition tool employed by the assistant fails to recognize Bong Joon-ho, it may provide an incorrect response. In such scenarios, it is expected that the assistant can learn this missing information about Bong Joon-ho and generalize it to other photographs. Thus, it is imperative to endow visual assistants with the learning capability, enabling them to swiftly acquire new knowledge from failures.

In this paper, we propose CLOVA, a Closed-LOop Visual Assistant that updates used tools via closed-loop learning [31, 33] to better adapt to new environments, as illustrated in Figure 1. CLOVA consists of three phases: inference, reflection, and learning. During inference, CLOVA employs Large Language Models (LLMs) to generate programs and execute corresponding tools to accomplish the task at hand. Subsequently, in the reflection phase, CLOVA utilizes human feedback to provide critiques, identifying tools that require updates. Finally, in the learning phase, CLOVA autonomously collects data and updates tools accordingly. Thus, CLOVA facilitates autonomous tool updating, thereby continually enhancing their ability to adapt to diverse environments.

To establish such a closed-loop learning framework, we must address three key challenges. Firstly, identifying tools that require updates is difficult due to the multi-step nature of generated programs and the diversity of errors within them. Secondly, automatically collecting training data is necessary as the knowledge to be learned is unpredictable. Thirdly, efficiently updating tools presents another obstacle, considering their scale and the quality of the collected data. Visual tools typically involve large neural networks, making them inefficient to update, and naive fine-tuning could result in unacceptable catastrophic forgetting [42]. Moreover, the presence of noise within the collected data further complicates the training process.

We propose several techniques to tackle these challenges. First, we introduce a multimodal global-local reflection scheme, which resorts to LLMs to identify tools that need to be updated from both global and local aspects. For the second challenge, three data collection manners are employed, including inferring answers by LLMs, searching on the Internet, and searching from open-vocabulary datasets. Lastly, we develop a training-validation prompt tuning scheme for the tools, which includes instance-wise prompt tuning and a subsequent prompt validation stage, where learned prompts that fail to predict the validation data will be discarded. The learning phase also updates LLMs by storing correct examples and incorrect examples with critiques as in-context examples, which will be used in future inference. As a result, CLOVA efficiently updates tools in challenging environments with noisy data, while avoiding catastrophic forgetting.

We apply CLOVA to compositional VQA and multiple-image reasoning tasks, using the GQA [19] and NLVRv2 [64] datasets. Additionally, we manually collect data for image editing and factual knowledge tagging tasks. CLOVA outperforms existing tool-usage methods by 5% in compositional VQA and multiple-image reasoning tasks, by 10% in knowledge tagging tasks, and by 20% in image editing tasks, showing the significance of the learning capability for general visual assistants.

In summary, our contributions are three-fold:

- We build CLOVA, a visual assistant that updates its tools within a closed-loop learning framework for better adaptation to new environments.
- We propose a multimodal global-local reflection scheme, capable of identifying tools in need of updates.
- We employ three flexible manners to automatically collect training data and introduce a novel training-validation prompt tuning scheme to update tools efficiently while avoiding catastrophic forgetting.

2. Related Work

2.1. General Visual Assistant

Benefiting from the advancements of LLMs [46, 67] and visual tools [24, 44, 55, 56], visual assistants have achieved great progresses. Some methods concatenate

Method	Visual Tool	Reflection	Update LLMs	Update VTs
ART [48]	✗	✗	Prompt	-
TRICE [53]	✗	Global	Instruction + RL	-
ToolkenGPT [12]	✗	-	✗	-
Toolformer [58]	✗	-	Fine-tune	-
VISPROG [10]	✓	✗	✗	✗
Visual ChatGPT [72]	✓	✗	✗	✗
HuggingGPT [60]	✓	✗	✗	✗
ViperGPT [65]	✓	✗	✗	✗
GPT4TOOLs [77]	✓	✗	Instruction	✗
OpenAGI [8]	✓	✗	RL	✗
AssistGPT [7]	✓	Global	Prompt	✗
CLOVA (Ours)	✓	Global+Local	Prompt	Prompt

Table 1. Comparisons with representative tool-usage methods, where VTs means visual tools.

and train LLMs with visual tools in an end-to-end manner, where representative work includes LLaVA [36], Otter [25], MMICL [80], Kosmos-2 [51], and Flamingo [1], *etc*. In addition, some work extends the idea of tool usage for AI assistants from natural language processing [4, 12, 48, 53, 54, 58] to computer vision. By providing in-context examples, VISPROG [10] and ViperGPT [65] generate programs to use visual tools. Following this idea, some work improves performance by collecting instruction-following data [37, 50, 77], adding more tools [34, 60], and designing more dedicated tool-usage procedures [15, 38–40, 72, 73, 78]. The most related work to CLOVA is AssistGPT [7] and OpenAGI [8]. The two methods update LLMs after development through in-context learning and reinforcement learning, respectively. Different from them, CLOVA can update both LLMs and visual tools via its reflection and learning phases. This allows CLOVA to better adapt to new environments. In addition, the closed-loop framework enables us to set a separate training stage for tool-usage methods, going beyond zero-shot or few-shot visual assistants. Comparisons between CLOVA and some representative tool-usage methods are shown in Tab. 1.

2.2. Reflection of LLMs

Reflection has become a remedy in case LLMs cannot generate good responses in a single attempt [5, 47, 49, 76]. Reflection methods send outputs back to LLMs to obtain critiques and further improve the outputs. These critiques take the form of scores or natural language [43, 45, 62, 83]. To generate better critiques, some methods employ instruction tuning [57, 75] or reinforcement learning [3, 53]. Recently, Huang *et al*. [17] revealed that LLMs struggle to provide accurate critiques for complex tasks. One way to address this issue is incorporating external information such as human-desired results into LLMs [2, 74, 76]. Unlike existing methods that rely solely on feedback in the language modality, our method generates reflection using all multimodal intermediate results. In addition, our method incorporates both global and local aspects for reflection, instead of only the global aspect. These bring more effective critiques for compositional tasks.

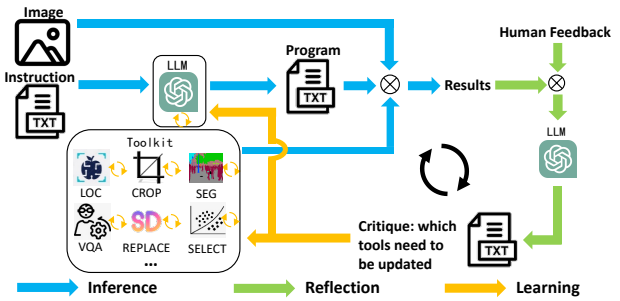

Figure 2. Framework of CLOVA.

2.3. Prompt-based Learning

Prompt-based learning is an efficient technique to update neural networks, achieving impressive performance in both NLP [21, 61] and computer vision [18, 59, 84, 85]. Prompt engineering and prompt tuning are two kinds of commonly used methods. Prompt engineering develops interpretable tokens (*e.g.*, texts and image regions) to guide model prediction, which are usually obtained by manually designing [55, 66], retrieval [79], and model generation [16, 52]. Prompt tuning learns vectors as prompts via gradient-based optimization. VPT [20] and CoOp [82] learn prompts for vision encoders and text encoders, respectively. To handle diverse data, CoCoOp [81] learns to generate prompts for unknown classes, ProDA [41] builds a Gaussian distribution for prompts, and MaPLe [23] learns both text and visual prompts. In addition, some methods employ prompts for continual learning, which learn prompts for different classes and produce adaptive prompts during inference. Representative methods include PIVOT [69], DualPrompt [70], and L2P [71]. Different from existing methods, our training-validation prompt tuning scheme discards harmful prompts, leading to more stable learning processes when the quality of training data is subpar.

3. Method

3.1. Overview

As shown in Fig. 2, CLOVA has three phases: inference, reflection, and learning. In the inference phase, CLOVA uses LLMs to generate programs and executes corresponding tools to solve the task. The reflection phase introduces a multimodal global-local reflection scheme that uses LLMs to generate critiques, identifying which tool needs to be updated. During learning, we employ three manners to collect training data and use a training-validation prompt tuning scheme to update the tools.

3.2. Inference

Our inference phase is based on VISPROG [10], while the difference is that CLOVA first uses LLMs to generate plans and then generates programs based on the plans, instead of

Tool Type	Tool Name	Tool Description	Data Collection
Tools to be updated	LOC	Use the OWL-ViT model [44] for object localization	Open-vocabulary datasest
	VQA	Use the BLIP model [27] for VQA	LLM inference
	SEG	Use the maskformer model [6] for panoptic segmentation	Open-vocabulary datasest
	SELECT	Use the CLIP model [55] to select the most relevant object, given a text description	Internet
	CLASSIFY	Use the CLIP model [55] to classify given images	Internet
	REPLACE	Use the stable diffusion inpainting model [56] to replace one object with another desirable object	Internet
Tools not to be updated	FACEDET	Use the DSFD model [26] for face detection	N/A
	LIST	Use the text-davinci-002 model of OpenAI for knowledge retrieval	N/A
	EVAL	Use the Python function eval() to process string expressions for answers	N/A
	RESULT	Use the Python function dict() to output the intermediate and final results	N/A
	COUNT	Use Python function len() to count the number of input bounding boxes or masks	N/A
	CROP	Use Python function PIL.crop() to crop images	N/A
	COLORPOP	Use Python function PIL.convert() to keep desirable objects in color and other regions gray	N/A
	BGBLUR	Use Python function PIL.GaussianBlur() to blur the background	N/A
	EMOJI	Use emojis in the Python packages AngLy(pypi) to hide someone's face	N/A

Table 2. Used tools in CLOVA, categorized based on whether the tool is updated in our method. Details of tool updates are in Sec. 3.4

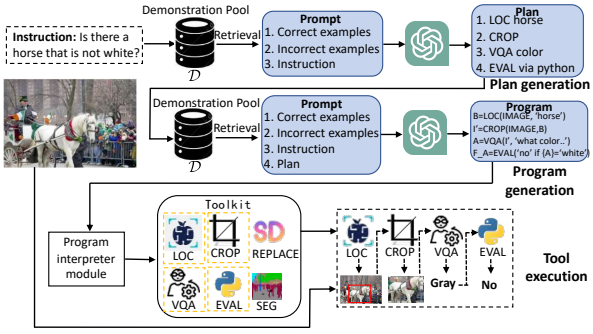

Figure 3. Illustration of the inference phase in CLOVA.

directly generating programs. Plans can be seen as intermediate reasoning chains that benefit the inference and reflection phases. Given a task, CLOVA selects in-context examples from a demonstration pool $\mathcal{D}$ (the construction of $\mathcal{D}$ will be detailed in Sec. 3.4.2), including correct examples and incorrect examples with error critiques. These examples are used to create prompts that are then sent to LLMs for plan and program generation. Finally, the program is parsed to execute visual tools (see Fig. 3).

Plan generation. The demonstration pool $\mathcal{D}$ is composed by $\mathcal{D} = \{\mathcal{D}_{p,s}, \mathcal{D}_{p,f}, \mathcal{D}_{c,s}, \mathcal{D}_{c,f}\}$, where $\mathcal{D}_{p,s}$ and $\mathcal{D}_{p,f}$ contain correct and incorrect examples for plan generation respectively, and $\mathcal{D}_{c,s}$ and $\mathcal{D}_{c,f}$ contain correct and incorrect examples for program generation respectively. Given a task, we use the BERT model [22] to extract features of the given instruction and examples stored in $\mathcal{D}_{p,s}$ and $\mathcal{D}_{p,f}$. Then, we combine similar examples in $\mathcal{D}_{p,s}$ and $\mathcal{D}_{p,f}$ with the instruction to create a prompt. Finally, we send the prompt to LLMs to generate the plan in a one-go manner.

Program generation. Similar to plan generation, we use LLMs to generate programs in a one-go manner. We select correct and incorrect examples of programs from $\mathcal{D}_{c,s}$ and $\mathcal{D}_{c,f}$. We combine these examples with the plan as a prompt and then send the prompt to LLMs for program generation.

Tool execution. We utilize the interpreter module in [10] to parse the program, extracting tool names, inputs, and outputs of each step. CLOVR activates tools from a toolkit

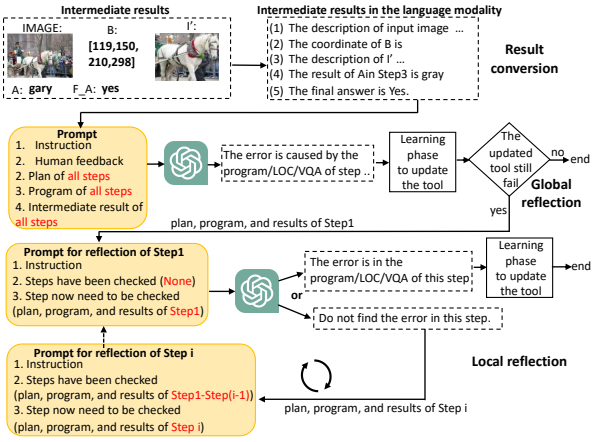

Figure 4. Illustration of the reflection phase in CLOVA.

$\mathcal{T}$ that contains 15 tools, including neural networks and Python functions, as shown in Tab. 2.

3.3. Reflection

In the inference phase, if a task is not solved correctly, the multimodal global-local reflection scheme uses LLMs to generate critiques, identifying which tool needs to be updated, as shown in Fig. 4.

Result conversion. Since LLMs often struggle to identify errors by themselves [17, 63, 68], we provide the human feedback, our wrong results, and intermediate results of each step for LLMs to better identify the error source. This requires us to convert visual results into textual form. For this purpose, we use the BLIP [27] model to convert images into languages by captioning.

Global reflection. CLOVA first uses global reflection to generate critiques in a one-go manner. The prompts are composed of task inputs, feedback on the task (*e.g.*, desirable results in VQA tasks, or human comments in image editing tasks), generated plans and programs, and intermediate results at each step. We send the prompts to LLMs to generate critiques that are used to update tools in the learning phase.

Local reflection. If CLOVA still fails after the tools are updated via global reflection and the learning phase–meaning the actual tools that lead to the faulty response are still to be found, we resort to local reflection to analyze each step of the program. Prompts are composed of the task inputs, feedback on the task, the steps that have been checked, and the current step that needs to be checked. Each step includes plans, programs, and intermediate results. We send the prompts to LLMs to infer whether this step has errors and the reasons. Local reflection continues until an error location and reasons are identified for a step.

3.4. Learning

3.4.1 Updating tools with prompt tuning

After identifying tools that need to be updated from the reflection phase, CLOVA then moves to the learning phase to collect training data and goes through training-validation prompt tuning to update the tools, as shown in Fig. 5.

Data collection. Since the tools that need to be updated can be rather different (a full list can be found in Tab. 2), we explore three manners to collect data online. (1) We use LLMs to generate training data for the VQA tool. If reflection concludes that the VQA tool makes errors, we combine the desirable response of the whole task and intermediate results of each step to prompt LLMs into inferring the correct output of the VQA tool. The question and the inferred output are then used to update the VQA tool. (2) We gather training data from open-vocabulary visual object grounding datasets (*e.g.*, LVIS [9]) for the LOC and SEG tools. For example, if the reflection phase indicates that LOC does not work well for the visual concept "*horse*", CLOVA will select images and bounding boxes of horses from LVIS to update LOC. (3) We collect data by searching on the Internet for the SELECT, CLASSIFY, and REPLACE tools. For instance, If CLASSIFY is marked as unable to recognize "*horse*" during the reflection phase, CLOVA will search the Internet for images of horses to update CLASSIFY.

Prompt tuning and validation. Given the collected data, we invoke training-validation prompt tuning to update tools. Note that, instead of learning a single prompt for all collected data, we choose to learn a prompt for each training instance collected. Each learned prompt will then be validated by running the tool with it on validation data (held out from collected data except for the VQA tool, where VQA will be validated on the original visual question it failed on) and seeing if the tool can produce desirable responses (*e.g.* correctly localizing a horse for the LOC tool). As a result, we discard prompts that do not lead to the desirable responses, possibly due to the faulty training instances they were trained on, alleviating the issue of noisy collected data. Finally, we build a prompt pool $\mathcal{P}$ for each tool. Take the LOC tool as an example. After training and validation, CLOVA stores the visual concept (*e.g.*, "horse" in the task of localizing horses) with its learned prompts and visual features of all collected instances in $\mathcal{P}$, formulated as $\mathcal{P} = \left\{v_j : \left[[f_{j1}, \cdots, f_{jn}], [p_{j1}, \cdots, p_{jn}]\right]\right\}_{j=1}^{m}$, where v_j is the name of the j-the concept (*e.g.*, "horse"), m concepts are stored totally. For the concept v_j, f_{ji} and p_{ji} are the feature and learned prompt using the i-th instance, respectively, and n instances are learned for v_j.

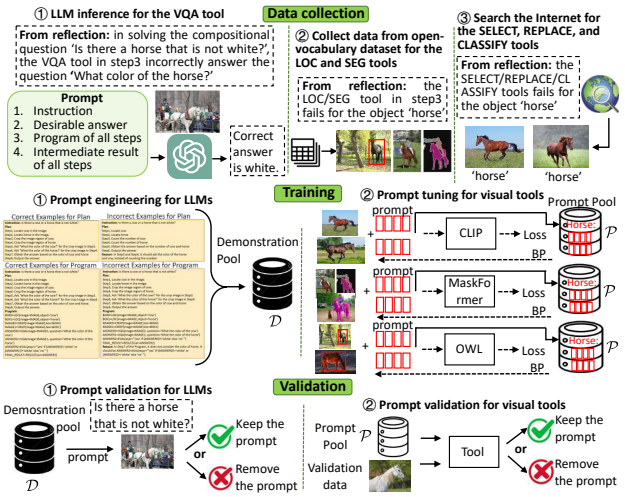

Figure 5. Illustration of the learning phase in CLOVA.

In summary, a tool is formulated as $T_{\theta,\mathcal{P}}$, where θ is the parameter of neural networks. The forward process of a sample x is $T_{\theta,\mathcal{P}}(x) = \theta([x, \mathcal{P}])$, where we concatenate x with a retrieved prompt from $\mathcal{P}$ as the input for the tool. We update the tool $T_{\theta,\mathcal{P}}$ by tuning $\mathcal{P}$ while θ being fixed,

$$\min_{\mathcal{P}} \mathbb{E}_{(x,y)} \mathcal{L}\big(T_{\theta,\mathcal{P}}(x), y\big),$$

where (x, y) is collected data, $\mathcal{L}$ is the loss function of $T_{\theta,\mathcal{P}}$.

Prompt ensemble. During inference, we use prompt ensemble to retrieve and utilize prompts from the learned prompt pool $\mathcal{P}$. Specifically, given a generated program, we first identify the visual concept for each involved tool in the program. For example, given an image editing task, "*Replace the dog with a cat*" where the SEG, SELECT, and REPLACE tools will be used, "*dog*" is the visual concept for the SEG and SELECT tools, and "*cat*" is the visual concept for the REPLACE tool. Then, in each step of tool usage with an input image x, if the visual concept is not in $\mathcal{P}$ of the corresponding tool, the prompt p' for x will be set as a zero-vector, *i.e.* the concept has not been learned for the tool so we just use the original tool (using zero-vector as a prompt); if the visual concept can be found in $\mathcal{P}$, *i.e.* the tool was updated with the visual concept before, we aggregate the prompts corresponding to this concept based on the similarity between features stored with these prompts and the feature extracted from the current input x: p' is computed by $p' = \frac{\sum_{i=1}^{n} w_i \cdot p_{ji}}{\sum_{i=1}^{n} w_i}$, where we compute the cosine similarity between feature f_x of x and features f_{ji} in $\mathcal{P}$ as the weight w_i.

3.4.2 Updating LLMs with demonstrations

Besides the visual tools, CLOVA can also update its LLMs. As we mentioned above, CLOVA utilizes a demonstration pool $\mathcal{D}$ to provide relevant examples for the LLMs. After working on new data, the plan, program, and reflection will be stored in $\mathcal{D}$ as correct or incorrect examples, based on whether the data is correctly proceeded. We also have a validation process that uses the original instruction as validation data to evaluate stored in-context examples. As the size of $\mathcal{D}$ grows, LLMs use more examples and therefore strengthen reasoning skills.

4. Experiments

4.1. Setting

Following VISPROG [10], we evaluate our method on four tasks: compositional VQA, multiple-image reasoning, language-guided image editing, and factual knowledge tagging, which requires visual perception, compositional reasoning, and image generation and manipulation abilities.

To comprehensively evaluate the learning capability of CLOVA, we set a separate training stage before deployment, which iteratively learns new knowledge via inference, reflection, and updating phases. In the test stage (*i.e.*, after development), we do not update LLMs and visual tools, and evaluate the performance only via the inference phase.

In the compositional VQA task, the GQA dataset [19] is used. We randomly select 500 samples from its train split as the training data, and 500 samples from its test-dev split as the test data. We report the top-1 accuracy. In the multiple-image reasoning task, we use the NLVRv2 dataset [64] that provides two images and a statement. We need to judge whether the statement is true or false. Similarly, we randomly select 500 samples from its train split as our training data, and 500 samples from its dev split as our test data.

Similar to VISPROG, we manually collect data for the language-guided image editing and factual knowledge tagging tasks. To better evaluate the learning capability, we collect fine-grained visual concepts that visual tools may not have learned, such as "*Replace the lion in the image with pine grosbeak*", where pine grosbeak is a fine-grained bird of Passeriformes. In the image editing task, we collect 129 images with 193 instructions, where 27 images with 78 instructions are used for training, and the rest are test data. We manually check whether edited images are semantically correct. The factual knowledge tagging task needs to identify persons or objects with bounding boxes in images. We collect 86 images with 177 instructions for this task, where 10 images with 88 instructions are used for training and the rest are used as the test data. We report the F1 score for this task.

In plan and program generation, prompts contain 4 correct examples and 4 incorrect examples. The demonstration pool $\mathcal{D}$ is initialized having about 20 in-context examples.

	Method	GQA	NLVRv2	Editing	Tagging
E2E	Otter [25]	48.2	48.2	-	-
	MMICL [80]	64.4	62.2	-	-
Tool	GPT4TOOLs [72]	41.2	45.4	17.8	
	Visual ChatGPT [72]	43.2	51.6	21.7	-
	InternGPT [38]	44.8	39.4	-	-
	HuggingGPT [60]	46.0	44.0	-	-
	ViperGPT [65]	47.2	-	-	-
	VISPROG [10]	49.8	60.8	40.2	0.393
	CLOVA (Ours)	**54.6**	**65.6**	**65.4**	**0.502**

Table 3. Comparisons in the four tasks. We report accuracies on GQA, NLVRv2, and image editing tasks, and F1 score on the knowledge tagging task. 'E2E' means end-to-end methods.

4.2. Main Results

We compare CLOVA with tool-usage methods: VISPROG [10], GPT4TOOLs [77], Visual ChatGPT [72], InternGPT [38], HuggingGPT [60], and ViperGPT [65]. We use their official codes, where all methods use the GPT-3.5 model. In addition, we also compare CLOVA with two advanced end-to-end models: Otter [25] and MMICL [80], which do well in GQA and NLVRv2 datasets. Results on the four tasks are shown in Tab. 3. We observe that CLOVA achieves the best performance among tool-usage methods. CLOVA has at least 4.8% improvements on the GQA dataset and 4.9% improvements on the NLVRv2 dataset. The reason is that CLOVA learns how to generate programs for the two tasks and update the VQA and LOC tools for better image perception. CLOVA performs competitively and even outperforms Otter and MMICL.

On image editing and knowledge tagging tasks, we do not compare our method with InterGPT, HuggingGPT, and ViperGPT, since they either need object masks or cannot accurately locate objects. In addition, most methods cannot finish the tagging task. Thus, we compare our method with VISPROG and Visual ChatGPT. As we collect find-grained data, it is challenging for off-the-shelf classification, segmentation, and image generation tools. Since GPT4TOOLs and Visual ChatGPT cannot use OpenCV functions and do not have the learning capability, they get bad performance on the image editing task. VISPROG can use OpenCV functions, but it cannot learn new knowledge. Its main fault is the inability to recognize or generate fine-grained concepts. Compared with them, the learning capability of CLOVA brings more than 20% and 10% improvements to image editing and knowledge tagging tasks, respectively.

4.3. Qualitative Results

In Fig. 6, we visualize four cases to illustrate the reflection and learning capability in CLOVA. It identifies tools that need to be updated, no matter LLMs or visual tools. Prompt engineering guides LLMs to generate correct programs for similar instructions. Visual tools learn new concepts via our

Figure 6. Case study of CLOVA on four example tasks.

data collection and prompt turning schemes. In Fig. 7, we visualize an example of global and local reflection. When the instruction is complex, the global reflection does not accurately identify which step has the error. Using global reflection as in-context examples still cannot generate correct programs. In contrast, local reflection successfully identifies the error step, and using local reflection generates correct programs, showing the effectiveness of local reflection.

4.4. Ablation Studies

We conduct ablation studies on the reflection and learning phases, using the GQA and NLVRv2 datasets. For reflection, we evaluate only using global reflection, only using local reflection, not using multimodal intermediate results, and not generating plans. We separately evaluate learning schemes for LLMs and visual tools. We evaluate only storing correct examples or incorrect examples for updating

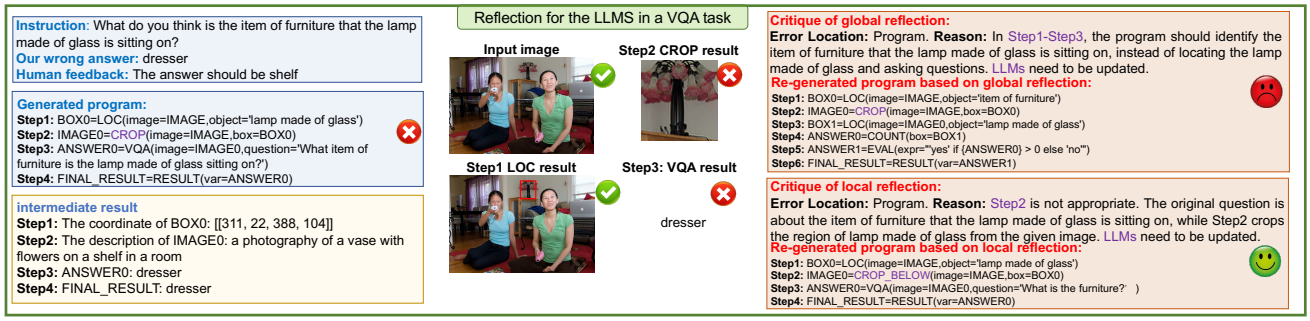

Figure 7. Visualization of the global reflection and local reflection in a VQA task.

	Method	GQA	NLVRv2
Reflection	w/o local reflection	52.0	65.2
	w/o global reflection	53.6	64.2
	w/o intermediate results	48.8	61.2
	w/o plan	50.0	62.6
	Ours	**54.6**	**65.6**
Prompt Engineering for LLMs	w/o incorrect cases	46.1	61.4
	w/o correct cases	48.2	63.2
	w/o validation	44.2	61.0
	Ours	**54.6**	**65.6**
Prompt Tuning for visual tools	w/o validation	42.8	62.8
	Ours	**54.6**	**65.6**

Table 4. Ablation on the GQA and NLVRv2 dataset.

Dataset	Method	LLaMA2-7B	GPT-3.5	GPT-4
GQA	Baseline	39.2	46.4	52.6
	+ Update LLMs	56.8	51.6	56.6
	+ Update visual tools	60.2	54.6	60.4
NLVRv2	Baseline	50.0	60.2	64.8
	+ Update LLMs	59.2	63.6	68.8
	+ Update visual tools	63.8	65.6	69.2

Table 5. Different LLMs on the GQA and NLVRv2 datasets.

LLMs. We also evaluate removing the validation process in prompt engineering and prompt tuning processes. Results are shown in Tab. 4. We find that these components are necessary for CLOVA to achieve better performance.

We evaluate CLOVA using different LLMs: LLaMA2-7B, GPT-3.5-turbo, and GPT-4. Results on GQA and NLVRv2 are shown in Tab. 5. We find that CLOVA leads to improvements in both strong LLMs (GPT-4) and weaker LLMs (GPT-3.5 and LLaMA2-7). We observe that CLOVA even achieves higher improvements on open-source LLMs (*i.e.*, LLaMA2-7B), 21% on the GQA dataset and 13.8% on the NLVRv2 dataset, bringing the significance of studying the learning capability of visual assistants. We further conduct experiments to evaluate CLOVA on two open-source LLMs: LLaMA2-7B and Mistral-7B. Results are shown in Tab. 6. We observe that CLOVA achieves significant improvements again.

5. Conclusion and Future Work

In this paper, we have presented CLOVA, a general visual assistant that can adapt to new environments via inference, reflection, and learning in a closed-loop framework. In the inference phase, using both correct and incorrect examples for prompts benefits to generate better plans and programs. Our reflection scheme is capable of identifying tools that need to be updated. Through three data collection manners and the validation-learning prompt tuning scheme in the learning phase, CLOVA can efficiently improve its tools. Experimental results on four tasks and different LLMs show the effectiveness of CLOVA as a general visual assistant with learning abilities.

Method	GQA	NLVRv2	Editing	Tagging
LLama2-7B	39.2	50.0	31.2	0.308
LLama2-7B + Ours	**60.2**	**63.8**	**47.6**	**0.357**
Mistral-7B	20.4	34.6	29.0	0.205
Mistral-7B + Ours	**31.4**	**42.2**	**46.5**	**0.303**

Table 6. Results on two open-source LLMs.

In the current method, we assume there is no selection or loop structure in programs, and assume there is at most one tool that needs updates in a task. The two assumptions cannot always hold in the real world. We could add selection and loop in-context examples for program generation and iterate the reflection and learning phases to update multiple tools. Besides, we could make some deployment designs to save the response time in our loop, including a foreground process and a background process. The former performs inference and gathers human feedback, while the latter does reflection, updates tools, and periodically synchronizes the weights of tools.

The framework of CLOVA can be easily generalized to new tools. (1) We will try multimodal LLMs (*e.g.*, LLaVA-1.5 [35]) to replace LLMs. In this case, we will evaluate the effectiveness of visual information for plan and program generation, reflection, and answer inference. (2) We can add more up-to-date tools (*e.g.*, BLIP2 [28] as a VQA tool and SAM [24] as an SEG tool), by just designing and programming their forward and prompt tuning processes.

Acknowledgements. We thank the anonymous reviewers for their constructive suggestions. Their insights have greatly improved the quality and clarity of our work. This work was supported in part by the National Science and Technology Major Project (2022ZD0114900).

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, pages 23716–23736, 2022.

[2] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *ICLR*, 2024.

[3] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.

[4] Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *ICLR*, 2024.

[5] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to self-debug. In *ICLR*, 2024.

[6] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, pages 17864–17875, 2021.

[7] Difei Gao, Lei Ji, Luowei Zhou, Kevin Qinghong Lin, Joya Chen, Zihan Fan, and Mike Zheng Shou. Assistgpt: A general multi-modal assistant that can plan, execute, inspect, and learn. *arXiv preprint arXiv:2306.08640*, 2023.

[8] Yingqiang Ge, Wenyue Hua, Jianchao Ji, Juntao Tan, Shuyuan Xu, and Yongfeng Zhang. Openagi: When llm meets domain experts. In *NeurIPS*, pages 5539–5568, 2023.

[9] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, pages 5356–5364, 2019.

[10] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *CVPR*, pages 14953–14962, 2023.

[11] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *CVPR*, pages 3608–3617, 2018.

[12] Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. In *NeurIPS*, pages 45870–45894, 2023.

[13] Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. Learning by fixing: Solving math word problems with weak supervision. In *Proceedings of the AAAI conference on artificial intelligence*, pages 4959–4967, 2021.

[14] Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. Smart: A situation model for algebra story problems via attributed grammar. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13009–13017, 2021.

[15] Pengbo Hu, Ji Qi, Xingyu Li, Hong Li, Xinqi Wang, Bing Quan, Ruiyu Wang, and Yi Zhou. Tree-of-mixed-thought: Combining fast and slow thinking for multi-hop visual reasoning. *arXiv preprint arXiv:2308.09658*, 2023.

[16] Xiaowei Hu, Zhe Gan, Jianfeng Wang, Zhengyuan Yang, Zicheng Liu, Yumao Lu, and Lijuan Wang. Scaling up vision-language pre-training for image captioning. In *CVPR*, pages 17980–17989, 2022.

[17] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *ICLR*, 2024.

[18] Siteng Huang, Biao Gong, Yulin Pan, Jianwen Jiang, Yiliang Lv, Yuyuan Li, and Donglin Wang. Vop: Text-video co-operative prompt tuning for cross-modal retrieval. In *CVPR*, pages 6565–6574, 2023.

[19] Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *CVPR*, pages 6700–6709, 2019.

[20] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *ECCV*, pages 709–727, 2022.

[21] Zhengbao Jiang, Frank F Xu, Jun Araki, and Graham Neubig. How can we know what language models know? *TACL*, 8:423–438, 2020.

[22] Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019.

[23] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *CVPR*, pages 19113–19122, 2023.

[24] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *ICCV*, pages 4015–4026, 2023.

[25] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023.

[26] Jian Li, Yabiao Wang, Changan Wang, Ying Tai, Jianjun Qian, Jian Yang, Chengjie Wang, Jilin Li, and Feiyue Huang. Dsfd: dual shot face detector. In *CVPR*, pages 5060–5069, 2019.

[27] 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, 2022.

[28] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pages 19730–19742, 2023.

[29] Qing Li, Jianlong Fu, Dongfei Yu, Tao Mei, and Jiebo Luo. Tell-and-answer: Towards explainable visual question answering using attributes and captions. *EMNLP*, 2018.

[30] Qing Li, Qingyi Tao, Shafiq Joty, Jianfei Cai, and Jiebo Luo. Vqa-e: Explaining, elaborating, and enhancing your answers for visual questions. In *ECCV*, pages 552–567, 2018.

[31] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *ICML*, pages 5884–5894. PMLR, 2020.

[32] Qing Li, Siyuan Huang, Yining Hong, and Song-Chun Zhu. A competence-aware curriculum for visual concepts learning via question answering. In *European Conference on Computer Vision*, pages 141–157. Springer, 2020.

[33] Qing Li, Yixin Zhu, Yitao Liang, Ying Nian Wu, Song-Chun Zhu, and Siyuan Huang. Neural-symbolic recursive machine for systematic generalization. *ICLR*, 2024.

[34] Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, et al. Taskmatrix. ai: Completing tasks by connecting foundation models with millions of apis. *arXiv preprint arXiv:2303.16434*, 2023.

[35] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *NeurIPS Workshop*, 2023.

[36] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, pages 34892–34916, 2023.

[37] Shilong Liu, Hao Cheng, Haotian Liu, Hao Zhang, Feng Li, Tianhe Ren, Xueyan Zou, Jianwei Yang, Hang Su, Jun Zhu, Lei Zhang, Jianfeng Gao, and Chunyuan Li. Llava-plus: Learning to use tools for creating multimodal agents. *2311.05437,arXiv*, 2023.

[38] Zhaoyang Liu, Yinan He, Wenhai Wang, Weiyun Wang, Yi Wang, Shoufa Chen, Qinglong Zhang, Yang Yang, Qingyun Li, Jiashuo Yu, et al. Interngpt: Solving vision-centric tasks by interacting with chatbots beyond language. *arXiv preprint arXiv:2305.05662*, 2023.

[39] Zhaoyang Liu, Zeqiang Lai, Gao Zhangwei, Erfei Cui, Zhiheng Li, Xizhou Zhu, Lewei Lu, Qifeng Chen, Yu Qiao, Jifeng Dai, and Wang Wenhai. Controlllm: Augment language models with tools by searching on graphs. *arXiv preprint arXiv:2305.10601*, 2023.

[40] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models. In *NeurIPS*, pages 43447–43478, 2023.

[41] Yuning Lu, Jianzhuang Liu, Yonggang Zhang, Yajing Liu, and Xinmei Tian. Prompt distribution learning. In *CVPR*, pages 5206–5215, 2022.

[42] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yuechen Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *ArXiv*, abs/2308.08747, 2023.

[43] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, pages 46534–46594, 2023.

[44] Matthias Minderer, Alexey Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, et al. Simple open-vocabulary object detection. In *ECCV*, pages 728–755, 2022.

[45] Varun Nair, Elliot Schumacher, Geoffrey Tso, and Anitha Kannan. Dera: enhancing large language model completions with dialog-enabled resolving agents. *arXiv preprint arXiv:2303.17071*, 2023.

[46] OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[47] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. *arXiv preprint arXiv:2308.03188*, 2023.

[48] Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. Art: Automatic multi-step reasoning and tool-use for large language models. *ArXiv*, abs/2303.09014, 2023.

[49] Joon Sung Park, Joseph C O'Brien, Carrie J Cai, Meredith Ringel Morris, Percy Liang, and Michael S Bernstein. Generative agents: Interactive simulacra of human behavior. In *UIST*, pages 1–22, 2023.

[50] Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *arXiv preprint arXiv:2305.15334*, 2023.

[51] 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.

[52] Sarah Pratt, Ian Covert, Rosanne Liu, and Ali Farhadi. What does a platypus look like? generating customized prompts for zero-shot image classification. In *ICCV*, pages 15691–15701, 2023.

[53] Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. Making language models better tool learners with execution feedback. *ArXiv*, abs/2305.13068, 2023.

[54] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shihao Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bowen Li, Ziwei Tang, Jing Yi, Yuzhang Zhu, Zhenning Dai, Lan Yan, Xin Cong, Yaxi Lu, Weilin Zhao, Yuxiang Huang, Junxi Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models. *arXiv preprint arXiv:2304.08354*, 2023.

[55] 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, 2021.

[56] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022.

[57] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing

models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

[58] Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *NeurIPS*, pages 68539–68551, 2023.

[59] Zhenwei Shao, Zhou Yu, Meng Wang, and Jun Yu. Prompting large language models with answer heuristics for knowledge-based visual question answering. In *CVPR*, pages 14974–14983, 2023.

[60] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in huggingface. In *NeurIPS*, pages 38154–38180, 2023.

[61] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, pages 4222–4235, 2020.

[62] Noah Shinn, Federico Cassano, Beck Labash, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, pages 8634–8652, 2023.

[63] Kaya Stechly, Matthew Marquez, and Subbarao Kambhampati. Gpt-4 doesn't know it's wrong: An analysis of iterative prompting for reasoning problems. In *NeurIPS Workshop*, 2023.

[64] Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In *ACL*, pages 6418–6428, 2019.

[65] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. In *ICCV*, pages 11888–11898, 2023.

[66] Jiajin Tang, Ge Zheng, Jingyi Yu, and Sibei Yang. Cotdet: Affordance knowledge prompting for task driven object detection. In *ICCV*, pages 3068–3078, 2023.

[67] 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.

[68] Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Can large language models really improve by self-critiquing their own plans? In *NeurIPS Workshop*, 2023.

[69] Andrés Villa, Juan León Alcázar, Motasem Alfarra, Kumail Alhamoud, Julio Hurtado, Fabian Caba Heilbron, Alvaro Soto, and Bernard Ghanem. Pivot: Prompting for video continual learning. In *CVPR*, pages 24214–24223, 2023.

[70] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *ECCV*, pages 631–648, 2022.

[71] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *CVPR*, pages 139–149, 2022.

[72] Chenfei Wu, Shengming Yin, Weizhen Qi, Xiaodong Wang, Zecheng Tang, and Nan Duan. Visual chatgpt: Talking, drawing and editing with visual foundation models. *arXiv preprint arXiv:2303.04671*, 2023.

[73] Jinjin Xu, Liwu Xu, Yuzhe Yang, Xiang Li, Yanchun Xie, Yi-Jie Huang, and Yaqian Li. u-llava: Unifying multi-modal tasks via large language model. *2311.05348,arXiv*, 2023.

[74] Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-seng Chua. Search-in-the-chain: Towards the accurate, credible and traceable content generation for complex knowledge-intensive tasks. *arXiv preprint arXiv:2304.14732*, 2023.

[75] Hao Yan, Saurabh Srivastava, Yintao Tai, Sida I Wang, Wentau Yih, and Ziyu Yao. Learning to simulate natural language feedback for interactive semantic parsing. In *ACL*, pages 3149–3170, 2023.

[76] Kevin Yang, Yuandong Tian, Nanyun Peng, and Dan Klein. Re3: Generating longer stories with recursive reprompting and revision. In *EMNLP*, pages 4393–4479, 2022.

[77] Rui Yang, Lin Song, Yanwei Li, Sijie Zhao, Yixiao Ge, Xiu Li, and Ying Shan. Gpt4tools: Teaching large language model to use tools via self-instruction. In *NeurIPS*, pages 71995–72007, 2023.

[78] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action. *arXiv preprint arXiv:2303.11381*, 2023.

[79] Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. What makes good examples for visual in-context learning? In *NeurIPS*, pages 17773–17794, 2023.

[80] 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. In *ICLR*, 2024.

[81] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16816–16825, 2022.

[82] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022.

[83] Pei Zhou, Aman Madaan, Srividya Pranavi Potharaju, Aditya Gupta, Kevin R. McKee, Ari Holtzman, Jay Pujara, Xiang Ren, Swaroop Mishra, Aida Nematzadeh, Shyam Upadhyay, and Manaal Faruqui. How far are large language models from agents with theory-of-mind? *arXiv*, abs/2310.03051, 2023.

[84] Jiawen Zhu, Simiao Lai, Xin Chen, Dong Wang, and Huchuan Lu. Visual prompt multi-modal tracking. In *CVPR*, pages 9516–9526, 2023.

[85] Muzhi Zhu, Hengtao Li, Hao Chen, Chengxiang Fan, Weian Mao, Chenchen Jing, Yifan Liu, and Chunhua Shen. Segprompt: Boosting open-world segmentation via category-level prompt learning. In *ICCV*, pages 999–1008, 2023.