Visual Document Understanding and Reasoning: A Multi-Agent Collaboration Framework with Agent-Wise Adaptive Test-Time Scaling

Supplementary Material

7. Related Works

7.1. General VLMs

Recent advancements in general VLMs have largely bridged the gap between visual and linguistic information, demonstrating impressive abilities in visual understanding, reasoning, and question answering. For example, latest GPT-4o [24], GPT-4V [44], Gemini-2.0-Pro [10], DeepSeek-VL2 [68], Qwen2.5-VL [2], Claude-3.7-Sonnet [1], InternVL-3 [88], Llama-3.2-Vision [18], Ovis2 [38] and MiMo-VL [71] contribute to push the performance boundaries in many aspects. However, there is still room for improvement for these general VLMs in terms of specific downstream tasks in various scenarios. Furthermore, their derivative models, such as LLaVA-OneVision [30], Monkey [33], Emu3 [63], InternLM-XComposer2-4KHD [11] and mPLUG-Owl3 [77], contribute to push the performance boundaries in many aspects.

7.2. Specialized VLMs

To better promote visual understanding and question answering, especially in document-based tasks, such as texts, webpages, charts, and tables, researchers have designed numerous task-specialized VLMs. For instance, TGDoc [65] and MMCA [34] are explorations of document-based or chart-based understanding via large-scale collected instruction tuning data, and DocPedia [15] deals with the visual inputs at the frequency domain. To improve the comprehension of screenshots and webpages, UReader [76] designs a cropping strategy, while CogAgent [21] and TextMonkey [36] use two resolution branches or a shifted window to accommodate document-oriented tasks. The series of works by mPLUG-DocOwl [22, 23] design a high-resolution compressor for multi-page document understanding with reduced visual token costs. Besides, ColPali [14], M3DocRAG [9], and MDocAgent [20] introduce efficient and novel retrieval mechanisms for VLVMs to comprehend long documents.

7.3. Multi-Agent Models

Agentic large models have gained tremendous popularity; however, single-agent systems are increasingly unable to meet the demands of complex and novel tasks. Thus, communicative agents' mechanisms are implemented to improve cooperation and reaction between agents by two primary forms, *i.e.*, debate [13, 45, 80, 81] and competition [6, 8]. Although the domain of LLM-based multi-agent models has expanded to include code generation [25, 56], financial decision-making [79], and other areas, they still suffer from a lack of visual understanding. Thus, more recently, some recent works attempt to explore how multi-agent interaction benefits VLMs. For instance, Insight-V [12] divides the functions on reasoning and summary agents, while Mobile-Agent-v2 [59] and [31] introduce specialized agents for planning. To accommodate different tasks, METAL [29], WebPilot [85], ViDoRAG [61], MatPlotAgent [74], and VipAct [86] further break down the labor of a single agent into more manageable sub-instances and specialise the function of each role-tailored agent. Additionally, MuMA-ToM [50] and MDocAgent [20] design separate image and text agents and then fuse them, ignoring the correlations between the two modalities. However, a practical multi-agent collaboration framework for visual document understanding and question answering is still deficient.

7.4. Test-Time Scaling

As a promising and burgeoning research focus, test-time scaling can further elicit various capabilities of models with relatively lower computational cost [83]. The scaling could be categorized into four types: parallel scaling [5, 35, 46, 58], generating multiple candidates and aggregating them; sequential scaling [7, 17, 78], iteratively updating intermediate states and triggering self-correction; Hybrid scaling [3, 19, 51] is another realization, which exploits their complementary benefits; internal scaling, which autonomously allocates computational tokens, *e.g.* budget forcing strategy [29, 43, 52]. To further enhance the ability of models to scale at test time, supervised fine-tuning (SFT) and reinforcement learning (RL) are also widely employed to improve scaling performance. For example, an intuitive and efficient strategy is to set a reward model, such as process reward modeling [51, 87] and outcome reward modeling [35, 48, 89], offering quantitative reward values for test-time scaling.

8. Methodology

8.1. Task Definition

For the question answering task defined in this work, given a question $\mathcal{Q}$ and the its corresponding visual document inputs $\mathcal{D} = \{v_1, v_2, \ldots, v_n\}$, the target is to generate a correct answer output $\mathcal{O}$. This answer should leverage the information within $\mathcal{D}$ and answer the proposed question compre-

Table 6. Preset tool library.

Name	Description	Calling Instruction
Search	Query external web or internal corpora and return ranked passages/documents for downstream reasoning.	call: search(query, top_k=10, source='web corpus', filters={}, dedup=True)
Retrieve	Vector/keyword retrieval over a prebuilt index; returns text chunks with metadata (id, score, source).	call: retrieve(index, query=None, ids=None, top_k=5, metric='cosine', namespace=None)
Captioning	Generate natural-language captions for images (or short videos) and optionally for specified regions.	call: caption(image_path, image_id, prompt=None, region=None, max_len=64)
Grounding	Ground a text query to image regions referring expression.	call: ground(image, text_query, top_k=5, return_bbox=True)
Translation	Translate text (and optionally OCR output) across languages.	call: translate(text—image_id, src='auto', tgt='en', preserve_format=True)
Code Interpreter	Execute sandboxed code for analysis, math, and plotting; returns stdout, errors, and produced files.	call: code_run(language='python', code, files=[], timeout=60, sandbox=True)
Region Cropping	Crop a rectangular/polygon region from an image and return the subimage and coordinates.	call: crop(image_path, image_id, bbox=[x_min, y_min, x_max, y_max] or points=[(x,y),...])
Webpage Reader	Read/parse HTML (optionally render JS) and select content.	call: page_read(url, selector=None, render_js=False)
PDF Reader	Read PDF pages as text/images/layout for mixed-modality docs.	call: pdf_read(pdf_id, pages='all', [1, 3, 5], return='text,images')

hensively and accurately.

8.2. Tool Library

As mentioned in Section 2.1, we equip $\mathcal{A}_{exe}$ with a tool library $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, to facilitate better execution and to dynamically endow models with specific capabilities. And each tool could be denoted as $t = \{name, desc, call\}$, where the three components are the name, description and the calling instruction of the tool. The tool library can also be expanded easily, allowing more flexible implementation. Namely, we preset commony used tools, as listed in Tab. 6.

For the tool selection, $\mathcal{A}_{exe}$ selects suitable tools for each task automatically. To be specific, we register the selected tool in the library in the form of a list in the prompt, as shown in Fig. 6.

8.3. Mixed Reward Modeling

As proposed in Section 2.3, we design a mixed reward, including both agent-specific and global outcome signals, to simultaneously optimize the multi-agent collaboration and further enhance the agent-specialized capabilities.

Agent-Specific Reward. For $\mathcal{A}_{plan}$ and $\mathcal{A}_{exe}$, whose outputs are step-wise, namely the output $\{s_1, s_2, \ldots, s_n\}$ and $\{e_1, e_2, \ldots, e_n\}$. Here, we use VisualPRM [62] as reward model $\mathcal{R}_{prm}$ to scalar reward given for single step output, as formulated:

$$r_{plan} = \mathcal{R}_{prm}\left(s_i \mid \mathcal{Q}, \mathcal{D}, \mathcal{P}_{context}\right), \tag{9}$$

$$r_{exe} = \mathcal{R}_{prm}\left(e_i \mid \mathcal{Q}, \mathcal{D}, \mathcal{E}_{context}\right), \tag{10}$$

where i denotes the i-th step of execution plan or execution process, while $\mathcal{P}_{context} = \{s_1, s_2, \ldots, s_{i-1}\} \subset \mathcal{P}$ and $\mathcal{E}_{context} = \{e_1, e_2, \ldots, e_{i-1}\} \subset \mathcal{E}$ are the contexts of the current step.

And for the $\mathcal{A}_{judg}$ and $\mathcal{A}_{ans}$, we utilize Skywork-VL-Reward [64] as output reward model $\mathcal{R}_{orm}$:

$$r_{judg} = \mathcal{R}_{orm}\left(\mathcal{J} \mid \mathcal{Q}\right), \tag{11}$$

$$r_{ans} = \mathcal{R}_{orm}\left(\mathcal{O} \mid \mathcal{Q}\right). \tag{12}$$

And we not only use the final selected answers but also include the other $N_p - 1$ rejected answers, providing denser reward signals and improving the generalization ability.

Global Outcome Reward. To enhance collaborative capabilities and mitigate the selfishness of each agent, we also employ a global outcome reward based on the final selected answer along the chosen path of the four agents, as follows:

$$r_{global} = \mathcal{R}_{orm}\left(\{\mathcal{P}_{sel}, \mathcal{E}_{sel}, \mathcal{J}_{sel}, \mathcal{O}_{sel}\} \mid \mathcal{Q}, \mathcal{D}\right). \tag{13}$$

where the set of $\mathcal{P}_{sel}, \mathcal{E}_{sel}, \mathcal{J}_{sel}, \mathcal{O}_{sel}$ represents the path on each agent of the final selected answer.

9. Experiments

9.1. Training Pipeline

In this work, we design a two-stage training pipeline to harness the full potential of our framework, specifically the SFT stage and RL stage. Moreover, we select three groups of base models for different variants, which are shown in Tab. 7. To enhance the general abilities, during the first stage, we employ AdamW optimizer with $\beta = (0.9, 0.95)$, and the initial learning rate and weight decay are set to $2e{-}5$ and 0.1 respectively. We train our model for one epoch with a batch size of 128. In the second RL stage, we adopt the GRPO [49] strategy for further optimisation. Following its original configurations, we set the learning rate, KL coefficient, and number of iterations as $e{-}6$, 0.04, and 1, respectively. The reward signals are generated by VisualPRM [62] and Skywork-VL-Reward [64]; the former provides step-by-step signals for planning and execution agents, while the latter generates rewards for judgment and answer agents. During the process, we set the temperature parameter to 0.75 to ensure diversity and creativity in the generated solutions. It should be noted that reward models and our framework are loaded on different servers, thereby avoiding the repeated loading and offloading of model weights. All the training and inference are conducted on 8 NVIDIA A100 GPUs.

9.2. Benchmarks

Our selected datasets cover four document types: text-based, webpage-based, table-based, and chart-based benchmarks, and also two non-document types: general and mathematical benchmarks. For each type, two to four datasets are included for both training and evaluation benchmarks. To better illustrate the datasets, statistical information provided in Tab. 8 and a brief introduction of each dataset are listed as follows:

- **DocVQA** [41]. It is a classic document-based dataset, with more than 12K images from various documents and 50K related questions. Each question corresponds to a single-page document, which requires relatively simple visual understanding. The average lengths of questions and answers are 8.3 and 2.1 words.
- **DUDE** [57]. It is a multi-page document question answering dataset, comprising 41,541 questions and over 5,000 documents. The average number of pages in the document of this dataset is 5.72, while the average lengths of the questions and answers are 8.7 and 3.4, respectively.
- **SlideVQA** [55]. It is a slide question answering dataset, which incorporates 52K images from slides and about 15K questions. The average lengths of the questions and answers are 9.0 and 2.4 words, and they include a certain proportion of multi-hop and numerical reasoning.
- **MMLongBench-Doc** [39]. It introduces a multi-page dataset for document understanding, consisting of about 6K document images and 1K questions. Depending on multi-hop evidence, the questions and answers have 16.4 and 2.8 words on average, and almost half of the questions are cross-page, while one-fifth of the questions are unanswerable.
- **VisualMRC** [54]. It collects 10K images sourced from 35 domains of webpages and more than 30K annotated questions with an average length of 10.6 words. The answers, with an average length of 9.5 words, in this dataset include both word spans in the context and the layouts of the webpages, which are more complex and diverse, with more abstract questions.
- **InfographicVQA** [42]. It consists of more than 5K images collected from webpages and 30K questions, constructed with textual, graphical, and visual elements. Most questions in this dataset are extractive and numerical, testing basic reasoning and arithmetic skills, with 11.5 and 1.6 words in questions and answers on average.
- **ChartQA** [40]. It introduces a dataset with 22K images and 34K questions, involving bar, line, pie charts, and composite plots. The questions in this dataset require visual, logical, and arithmetic ability, with an average length of questions and answers of 7.2 and 1.2 words.
- **CharXiv** [66]. It presents a dataset for chart understanding and answering, with 2K chart images and 13K curated questions. It includes both descriptive and reasoning subdatasets, and we merge them. The questions and answers have 22.6 and 2.8 words evenly.
- **TableBench** [67]. It is a dataset with tabular and textual hybrid contents, constructed with over 4K tables and 21K questions, with numerical reasoning, such as computing, comparison, and sorting. Its questions and answers have lengths of 20.3 and 8.5 words on average.
- **TableVQA-Bench** [26]. It establishes a dataset for table-based question answering, extended with four existing sources with 1K tables and 2K questions. Consisting of diverse types of questions, the average lengths of questions and answers are 10.5 and 4.7 words.
- **ScienceQA** [47]. It is a dataset that includes about 21K multiple-choice questions based on more than 10K images from diverse scientific fields, and the length of the questions and answers is 12.1 words on average.
- **RealWorldQA** [69]. It consists of about 1K questions and 1K images. The average length of the questions in this dataset is 11.1 words, while that of the answers is only 1.2 words.
- **MathVista** [37]. It proposed a dataset integrating diverse mathematical and visual tasks with 5K images and 6K questions, demanding fine-grained visual understanding and compositional reasoning. On average, the length of the questions and answers is 15.6 and 1.2 words, respectively.

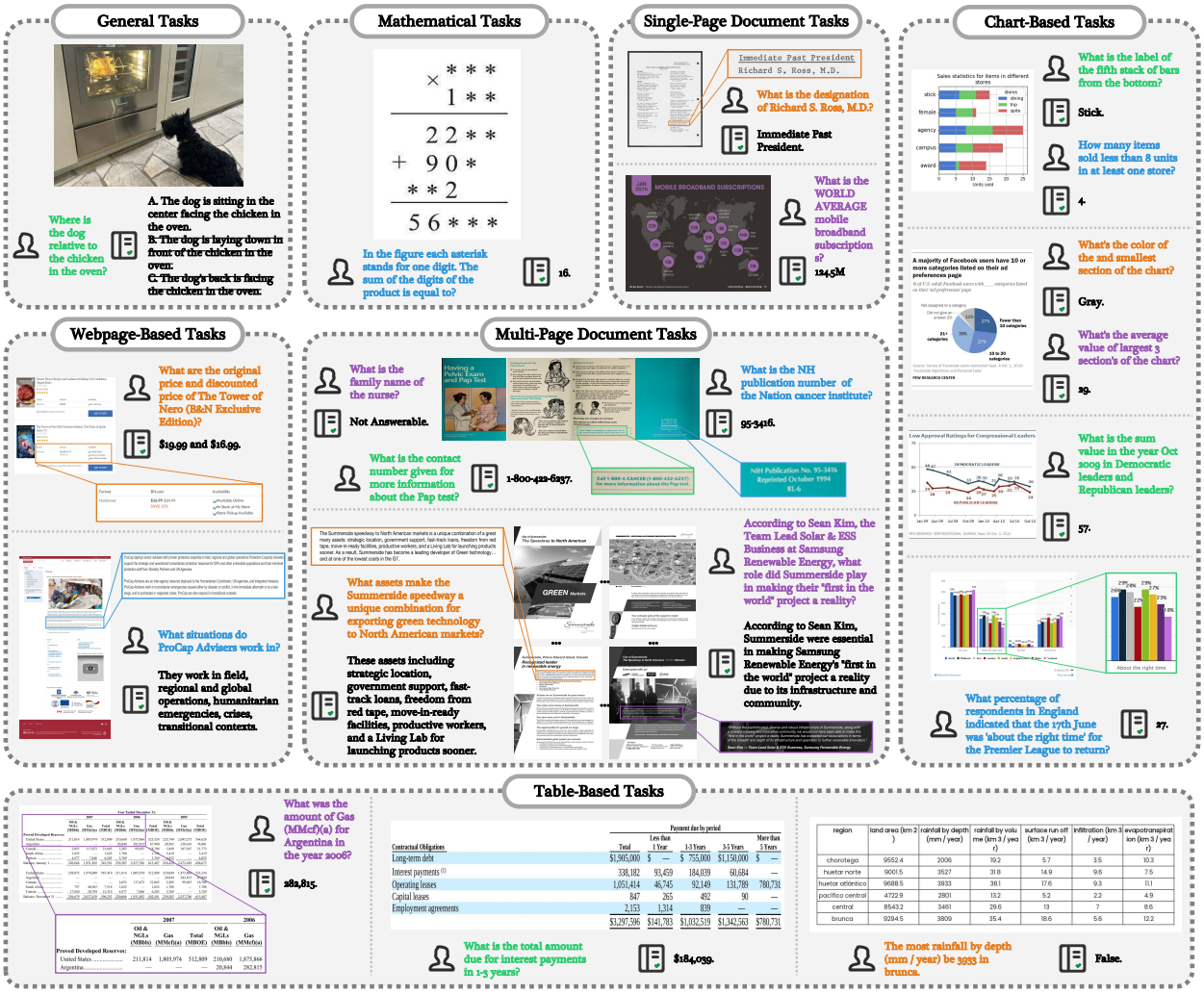

Figure 5. Illustrations of representative samples from our selected 15 benchmarks on both document and non-docuement tpyes.

Table 7. The base models of each variant of MACT. * indicates that we use VLMs, which remain powerful textual ability, because in InternVL3 series the model family does not provide LLMs.

	Type	Qwen2.5-VL Series [2, 73]	MiMo-VL Series [70, 71]	InternVL3 Series [88]
Planning Agent	VLM	Qwen2.5-VL-7B-Instruct	MiMo-VL-7B-SFT	InternVL3-9B-Instruct
Execution Agent	VLM	Qwen2.5-VL-7B-Instruct	MiMo-VL-7B-SFT	InternVL3-9B-Instruct
Judgment Agent	LLM*	Qwen2.5-7B-Instruct	MiMo-7B-SFT	InternVL3-8B-Instruct
Answer Agent	LLM*	Qwen2.5-3B-Instruct	MiMo-7B-SFT	InternVL3-3B-Instruct
Size	-	7B+7B+7B+3B =24B	7B+7B+7B+7B =28B	9B+9B+8B+2B =28B

- **Math-Vision** [60]. It is a dataset integrating diverse mathematical tasks with about 3K images and 3K questions from real exams, spanning 16 distinct mathematical disciplines and graded across 5 levels of difficulty. It has the longest average length of questions (42.3 words) in our selected datasets, and 2.2 words in answers on average.
- **MathVerse** [28]. It is a mathematical dataset with about

Table 8. Statistical information of our selected datasets. Here, Num and Avg indicate the total number and the average length, while I, Q, and A represent images, questions, and answers, respectively.

Dataset	Type	Num I	Num Q	Avg Q	Avg A	Avg I Per Q
DocVQA [41]	Text	12K	50K	8.3	2.1	1.0
DUDE [57]	Text	5K	41K	8.7	3.4	5.7
SlideVQA [55]	Text	52K	15K	9.0	2.4	20.0
MMLongBench-Doc [39]	Text	6K	1K	16.4	2.8	47.5
VisualMRC [54]	Webpage	10K	30K	10.6	9.5	1.0
InfographicVQA [42]	Webpage	5K	30K	11.5	1.6	1.2
ChartQA [40]	Chart	22K	34K	7.2	1.2	1.0
CharXiv [66]	Chart	2K	13K	20.4	2.9	1.0
TableVQA-Bench [26]	Table	1K	2K	10.5	4.7	1.0
TableBench [67]	Table	21K	4K	20.3	8.5	1.0
ScienceQA [47]	General	10K	21K	12.1	4.4	0.5
RealWorldQA [69]	General	1K	1K	11.1	1.2	1.0
MathVista [37]	Mathematical	5K	6K	15.6	1.2	1.0
Math-Vision [60]	Mathematical	3K	3K	42.3	2.2	1.0
MathVerse [84]	Mathematical	3K	16K	35.7	1.4	1.0

3K images and 16K questions. On average, the lengths of the questions and answers in this dataset are 35.7 and 1.4 words, respectively.

For some datasets, we use pre-treatments to handle the visual inputs. For the document in some datasets, which is provided in PDF format, we transform these files into visual images as input. Besides, we design a standardised visualisation process for part of the tables in the TableBench [67] dataset, which only contains HTML format data without authentic visual images. Additionally, we utilize the external PDF2Image tool to convert PDF files into images.

9.3. Evaluations

We onboard the unavailable models in LMMs-Eval [82] to evaluate their performances with the consistent configurations. We directly use the natively supported benchmarks in the LMMs-Eval framework, namely DocVQA [41], InfographicVQA [42], ChartQA [40], ScienceQA [47], RealWorldQA [69], MathVista [37], and MathVerse [84]. In terms of unavailable benchmarks in LMMs-Eval, namely DUDE [57], SlideVQA [55], MMLongBench-Doc [39], VisualMRC [54], CharXiv [66], TableVQA-Bench [26], TableBench [67], and Math-Vision [60], we onboard the benchmarks and register them by YAML configurations.

To evaluate the performance of models on these benchmarks fairly and equitably, we employ a range of evaluation metrics. For DocVQA [41], DUDE [57], InfographicVQA [42] benchmarks, we follow the original evaluation metrics, *i.e.*, ANLS (Average Normalised Levenshtein Similarity) [4], to measure the difference between the predicted answers and ground truths. Moreover, the F1 score is utilized for SlideVQA [55]. For other selected benchmarks, we use powerful GPT-4o [24] as a judge model to evaluate the answers by LMMs-Eval, which generates scores from 1 to 10, and then the scores will be scaled to an accuracy between 0 and 100 in the dimension of benchmarks.

9.4. Prompt Templates

As shown in Fig. 6, there are prompts for the four agents, including the system prompts and the prompts for each process. Apart from the system prompts, $\mathcal{A}_{plan}$ and $\mathcal{A}_{exe}$ have two extra prompts to accomplish their functions, while $\mathcal{A}_{judg}$ and $\mathcal{A}_{ans}$ have only one. In the separating prompt of $\mathcal{A}_{exe}$, each execution unit is comprised of: (1) a specific definition; (2) expected target and output; (3) existing inputs or results from the previous step.

10. Results and Discussions

10.1. Additional Quantitative Results

For the line graphs and bar charts shown in Fig. 4a and the line graphs in Fig. 4b, we list the quantitative results in Tab. 9 and Tab. 10, respectively, for clarity. Compared with the baseline, our judgment agent strategy for correction improves the scores by 6.2% on average when the maximum number of corrections is set to 3. Furthermore, it still has 5.3% and 2.6% increases when compared to the other two correction strategies. Besides, our strategy achieves the best performance with fewer average number of corrections, indicating high efficiency and accuracy. In terms of the impact of the number of generated plans N_p and the number of candidate executions N_e, it is evident that higher values

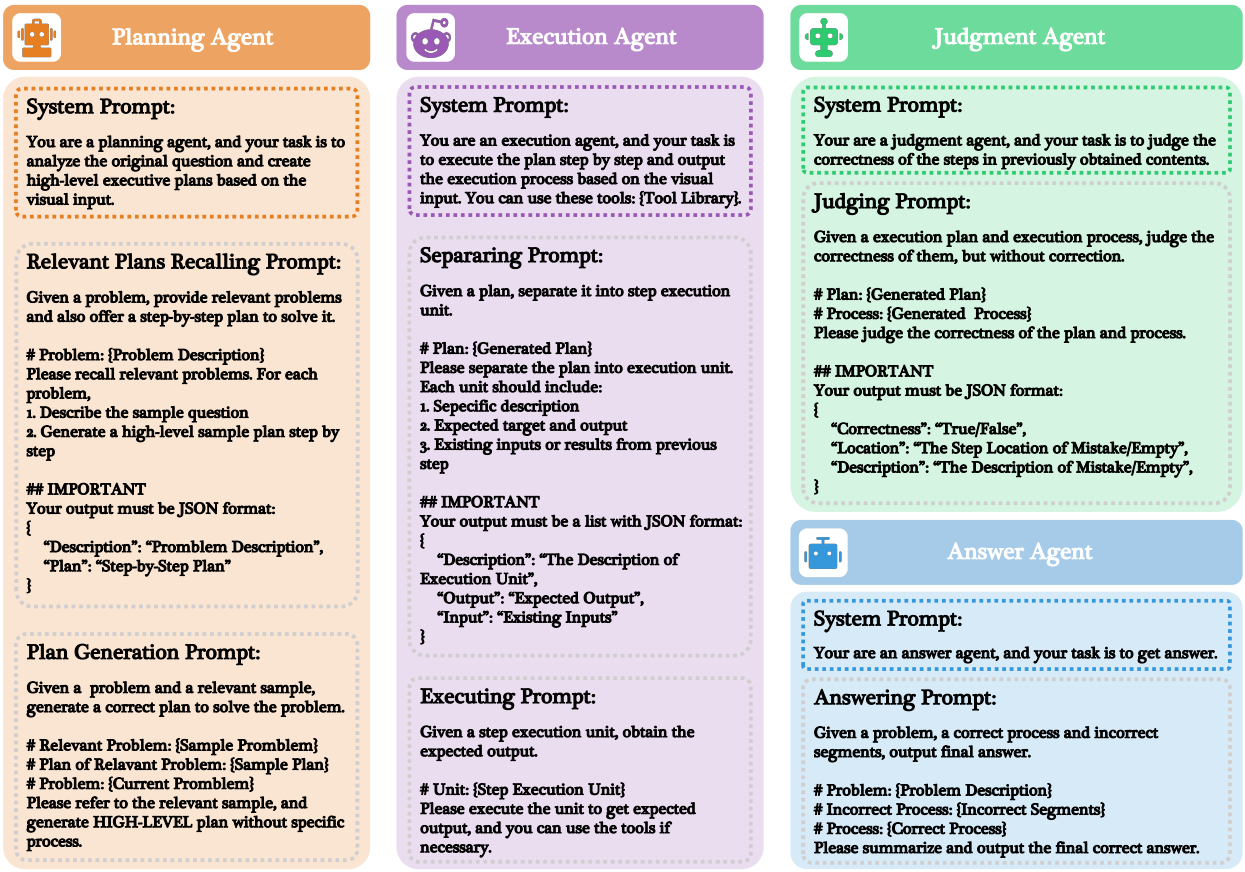

Figure 6. Prompts for the four agents.

Table 9. Comparisons among the three correction strategies of various maximum numbers of corrections and the average number of corrections when the limitation is set to 3. Line with * denotes the results on the three selected benchmarks, while another line is the average results on all benchmarks. "Cor." means correction.

	Max Num of Cor.				Avg
	3	5	7	10	Cor.
Baseline*	40.2				-
Internal Correction*	40.8	42.0	41.5	38.2	2.8
Agent for Judgment and Correction*	45.1	46.7	44.6	41.2	2.3
Independent Judgment Agent (Ours)*	47.6	47.5	45.3	42.9	1.9
Baseline	68.8				-
Internal Correction	69.5	70.3	69.6	64.2	2.3
Agent for Judgment and Correction	72.2	72.6	71.9	68.9	1.6
Independent Judgment Agent (Ours)	74.8	74.7	73.7	71.2	1.3

of the two parameters cause superior performance.

Table 10. Results of different values of N_p and N_e on all benchmarks.

N_p \ N_e	1	2	4	8	16
1	70.2	70.8	72.0	72.5	72.8
2	71.1	-	-	-	73.4
4	71.5	-	-	-	73.9
8	72.0	-	-	-	74.5
16	72.2	72.9	73.4	74.1	74.8