Front. Comput. Sci., 2025, 19(11): 1911367
https://doi.org/10.1007/s11704-025-41244-0

RESEARCH ARTICLE

CCA: collaborative competitive agents for image editing

Tiankai HANG[1], Shuyang GU[2], Dong CHEN[2], Xin GENG (✉)[1], Baining GUO (✉)[1,2]

1 School of Computer Science and Engineering, Southeast University, Nanjing 211189, China
2 Microsoft Research Asia, Beijing 100080, China

Abstract This paper presents a novel generative model, Collaborative Competitive Agents (CCA), which leverages the capabilities of multiple Large Language Models (LLMs) based agents to execute complex tasks. Drawing inspiration from Generative Adversarial Networks (GANs), the CCA system employs two equal-status generator agents and a discriminator agent. The generators independently process user instructions and generate results, while the discriminator evaluates the outputs, and provides feedback for the generator agents to further reflect and improve the generation results. Unlike the previous generative model, our system can obtain the intermediate steps of generation. This allows each generator agent to learn from other successful executions due to its transparency, enabling a collaborative competition that enhances the quality and robustness of the system's results. The primary focus of this study is image editing, demonstrating the CCA's ability to handle intricate instructions robustly. The paper's main contributions include the introduction of a multi-agent-based generative model with controllable intermediate steps and iterative optimization, a detailed examination of agent relationships, and comprehensive experiments on image editing.

Keywords image editing, agents, collaborative and competitive

1 Introduction

The human endeavor to conceptualize Artificial Intelligence (AI) is fundamentally rooted in the aspiration to engineer intelligent entities. In the contemporary era, this pursuit has been significantly propelled by the evolution and advancement of Large Language Models (LLMs) [1–5]. The rapidly developing LLM-based agents [6–8] have outpaced their predecessors, demonstrating a higher degree of intelligence through a more sophisticated understanding of human intentions and a greater competency in assisting with complex tasks. This progression has instigated a technological revolution across a multitude of fields, encompassing software development [7], education [8,9], sociology [10], among others.

Received November 17, 2024; accepted February 13, 2025

E-mail: xgeng@seu.edu.cn; 307000167@seu.edu.cn

When examining the realm of generative models, specifically Generative Adversarial Networks (GANs) [11–14] and diffusion models [15–22], we encounter two notable challenges. The first challenge is the models' limited ability to process complex, compound tasks. To illustrate, consider a task that involves "colorizing an old photograph, replacing the depicted individual with the user's image, and adding a hoe in the user's hand". Such a multifaceted task surpasses the capability of even the most advanced generative models. The second challenge arises in the update process of a generated result. This process is contingent upon the preservation of the compute graph. However, the sheer volume of results generated by diverse algorithms makes maintaining this compute graph a significant hurdle. Consequently, this creates a barrier to learning from other generative models, given their black-box nature.

In this paper, we introduce a novel generative model that harnesses the capabilities of multiple LLM-based agents, which effectively circumvents these two challenges. Leveraging the agents' powerful task decomposition abilities, our model can efficiently manage highly complex tasks. Simultaneously, during the generation process, we can extract insights into how the agents comprehend, dissect, and execute the task, enabling us to modify internal steps and enhance the results. Crucially, the model's transparency allows the agent to learn from successful executions by other agents, moving away from the black-box model paradigm. We underscore that this transparency is a pivotal factor contributing to the enhanced quality and robustness of the system.

Generative Adversarial Networks (GANs) [11] can be viewed as an early endeavor to incorporate a multi-agent system into generative models. GANs utilize two agents, namely, a generator and a discriminator. A cleverly designed optimization function allows these agents to learn from their adversarial interaction, ideally reaching a Nash equilibrium. Similarly, in our multi-agent system, we have discovered that the establishment of relationships between different agents is a critical determinant of success.

Drawing inspiration from GANs, our system employs two generators and one discriminator. The two generator agents, of equal status, independently process user instructions and generate results. The discriminator agent then evaluates these generated results, providing feedback to each generator and

determining which result is superior. The generator agents have dual responsibilities. Firstly, they must reflect on the feedback from the discriminator. Secondly, they should consider the results produced by the other generator agent to enhance their generation process. This process is iteratively carried out until the discriminator deems the best result to have sufficiently met the user's requirements. We underscore that through this collaborative competition, the two generators can continuously augment the quality and robustness of the system's results. Consequently, we have named our system Collaborative Competitive Agents (CCA).

In this paper, we concentrate on image editing, although our CCA system is a versatile generative model. Conventional image editing methods [23–25] fall short when dealing with intricate instructions, resulting in less robust outcomes. Our proposed generative model can considerably enhance this situation through the collaborative competition of multiple agents.

In summary, our primary contributions are as follows:

(1) We introduce a new generative model based on multiple agents, which features controllable intermediate steps and can be iteratively optimized.
(2) We have meticulously examined the relationships among multiple agents, highlighting that reflection, cooperation, and competition are integral to the system's quality and robustness.
(3) We have conducted comprehensive experiments on image editing, demonstrating for the first time the ability to robustly handle complex instructions.

2 Related work

2.1 Large language model-based agents

Agents are artificial entities capable of perceiving the environment, making decisions, and taking actions to accomplish specific goals [26–29]. Recent advancements in Large Language Models (LLMs) have demonstrated significant intelligence [30,31], offering promising avenues for the evolution of intelligent agents. LLM-based agents possess the ability to memorize, plan, and utilize tools. The "memory" feature allows these agents to store sequences of past observations, thoughts, and actions for future retrieval. The CoT [32] enhances the LLM's capacity to solve complex tasks by "thinking step by step". Moreover, agents employ a reflection mechanism [33–35] to enhance their planning abilities. Furthermore, LLM-based agents can leverage various tools to interact with their environment [2,30,36], such as shopping [6] and coding [7]. Some studies [37] equip these agents with embodied actions to facilitate interaction with humans.

Similar to human society, a single skilled agent can handle specific tasks, while a multi-agent system can tackle more complex ones. To foster autonomous cooperation among agents, CAMEL [38] introduces a novel communicative agent framework called "role-playing", which incorporates inception prompting. AgentVerse [39] introduces a multi-task-tested framework for group agents collaboration, designed to assemble a team of agents that can dynamically adjust to the intricacy of the task at hand. ChatDev [7] demonstrates significant potential in software development by integrating and unifying key processes through natural language communication. Concurrently, ChatEval [40] employs multiple agents as a referee team, they engage in debates among themselves and ultimately determine a measure of the quality of the LLM generation. Pure collaboration means that agents work together, completing their respective parts to achieve a common goal. Competition, on the other hand, implies rivalry, where each agent's objective is to pursue their own success, making their own plans and decisions based on feedback from both themselves and other agents. Therefore, this is not a situation of complete cooperation. We refer to this type of competition as collaborative competition. In this paper, we explore a scenario where multiple agents with collaborative competition to achieve goals.

2.2 Image editing

Image Editing has been a long-standing vision task and is widely used in real-world applications [23,24,41–43]. The primary objective of this task is to manipulate an image to align with user-specified requirements. Traditional methods mainly address specific tasks like style transfer [44–47], image translation [48,49], and object removal or replacement [50–53]. Later works [41,54] utilize text-to-image models to perform edits from user-provided text instructions.

From the generative model perspective, many early studies leveraged the outstanding disentangled properties in the latent space of GANs [13,14], altering image attributes via latent code manipulation [55–57]. Some studies [58] use CLIP [59] to facilitate text-driven image editing. Recent advancement in diffusion models [15–18,60] has demonstrated great success in image generation [61–64], with these pre-trained diffusion models widely used in image editing. Meng et al. [25] proposed reversing and perturbing the SDE to conduct image manipulation, while EDCIT [65] enhanced editing quality through more precise inversion. Prompt-to-Prompt [24] investigated the role of words in attention and edited images by modifying the cross-attention. Null-text Inversion [43] progresses by optimizing tokens to avoid artifacts from classifier-free guidance. Dreambooth [66] fine-tuned the pre-trained text-to-image diffusion model to perform subject-driven generation. More recent work, VisProg [67], leverages the *in-context learning capabilities* of LLMs to perform image editing. However, it*heavily relies on the given examples* and choices for the most suitable tool. In contrast to it, our proposed CCA is a multi-agent system that not only includes tools for planning and execution but also iteratively improves the outcomes based on feedback through collaboration and competition.

3 Method

Our framework, identified as CCA, incorporates two distinct types of agents: the Generator Agent and the Discriminator Agent.

3.1 Generator agent

The generator agent edits the image through the utilization of two core modules: the **Planner**, which is engaged in

deciphering the user's request and making plans, and the **Tool Executor**, responsible for systematically modifying the image in a step-by-step manner.

3.1.1 Planner

Existing image editing models often grapple with effectively managing complex user requirements. To address it, we employ a Planner agent, denoted as $\mathcal{P}$, to decompose these requirements into several straightforward and clear subtasks. Typically, user requirements encompass an input image I and an associated editing goal, G. To enable the Large Language model to comprehend the image, we utilize LLaVA-1.5 [68] to get the image caption C, which serves as a preliminary understanding of the input image. For simplicity, We denote these three elements as user requirements, R.

The Planner agent $\mathcal{P}$ takes the requirements R as inputs, decomposing them into a sequence of subtasks, represented as $\{s_j\}_{j=1}^{n}$.

$$\{s_1, s_2, \ldots, s_n\} = \mathcal{P}(R, T). \tag{1}$$

Each subtask s_j contains a clear goal to be achieved and a selected tool from Toolset T to accomplish this goal. The toolset outlines which tools are available to the agent and their respective functions, which will be detailed further in Section 4.1.

For instance, if the user request is "*Make the background a county fair and have the man a cowboy hat, 512pix*". It can be divided into several sequential steps: 1. Subtask s_1 involves loading and resizing the image to a resolution of 512 pixels using the `Resize` tool; 2. Subtask s_2 requires Changing the background to a county fair using the editing tool `Edict` [65]; 3. Lastly, subtask s_3 requires adding a cowboy hat to the man using the `InstructDiffusion` [41] tool.

It's plausible that multiple tools could handle the same subtask, but each tool may have distinct advantages in different scenarios. Consequently, generating an optimal plan on the first attempt can be challenging. In response to this, we have introduced a strategy that enhances the planning process through multiple rounds of reflection. The *reflection* mechanism is designed to incrementally improve the plan to meet the editing requirements by utilizing feedback. The feedback F assesses the success of achieving a sub-goal with the chosen tool and determines whether modifications to the sub-goal are necessary. The feedback is generated by the discriminator agent, which will be further discussed in Section 3.2.1.

In summary, beyond its primary function, the Planner agent also serves to reflect upon and enhance plans based on the preceding plan:

$$S^m = \mathcal{P}(R, T, S^{m-1}, F^{m-1}), \tag{2}$$

where the superscript m denotes the current round of plan. In the initial round, when m equals 1, there are neither feedbacks nor previous plans available, thus both $S^{(0)}$ and $F^{(0)}$ are set to $\emptyset$ (empty).

3.1.2 Executor

When the Planner agent generates a detailed plan that specifies which tool should be employed for each task, we engage another agent, the Executor $\mathcal{E}$, to use the corresponding tools to sequentially execute the plan $\{s_1, s_2, \ldots, s_n\}$. For each individual subtask s_j, the Executor should meticulously explore how to optimally leverage the tool to accomplish it.

For its initial run, the Executor should carefully review the tool's detailed instructions, and appropriately format the input to engage the tool. In subsequent runs, the executor may receive feedback on previous execution results, then it should adjust the hyperparameters according to these previous results to enhance future outcomes. The entire process can be formulated as follows:

$$o_{j+1} = \mathcal{E}(s_j, o_j, f_j), \quad j = 1, 2, \ldots, n. \tag{3}$$

In this process, o_j and f_j represent the previously generated results and feedback, respectively. The system output is defined as $O = o_{n+1}$.

For instance, during the initial run, the Executor may not effectively "add a hat" due to the use of inappropriate classifier-free guidance. In response to the feedback signal "the hat has not been added", the Executor may enhance the classifier-free guidance to improve performance.

3.2 Discriminator agent

Evaluating the results is a crucial step towards their improvement. Hence, we employ a Discriminator Agent which serves a dual purpose. Firstly, it is responsible for assessing the quality of the edited images and providing valuable feedback that contributes to the enhancement of these results. Secondly, it is tasked with selecting the best results to present to the user.

3.2.1 Generate feedback

Given the caption C of the input image and the user request R, we can design several questions to assess whether the generated image meets the stipulated requirements. Questions assess specific request items and overall editing quality in two parts. The first part verifies if edits match user requirements, typically prompting a binary "Yes/No" and explanation. The second part rates overall quality, considering naturalness and aesthetics.

The agent sequentially answers these evaluative questions. For questions concerning specific editing items, the agent enlists help from LLMs with visual question-answering capabilities, such as LLaVA [68] or GPT-4V [2]. These models take the edited image and question as input and output the answer. As for the comprehensive quality assessment, we not only rely on these types of LLMs, but we also incorporate the Aesthetic Predictor [69] to evaluate the naturalness and visual appeal of the output.

We compile responses into a succinct feedback report. The feedback offers clear, actionable directions for editing enhancement. The entire process can be formalized as follows,

$$F = \mathcal{FB}(O, R), \tag{4}$$

where $\mathcal{FB}(\cdot)$ is the agent to generate feedback F, and O denotes the output from the generator.

To enhance the generator's ability to reflect and improve

through feedback, we undertake two measures: Firstly, we transmit the feedback from both generator agents back to their respective origins, enabling them to learn from each other's successful strategies or avoid redundant exploration. Secondly, we dissect the overall feedback for each subtask s_j. This process enables the planner to more effectively discern the appropriateness of the goal and tools with each subtask. Additionally, the specific feedback guides the executor in fine-tuning hyperparameters, even with unchanged goals.

Consider the first generator agent in Fig. 1, in the first round, the feedback is "A rainbow is visible in the sky, there are no sunflowers in the field, and there is no indication of a wooden barn in the image. Unpleasant visual artifacts are present in the photo". The decomposed feedback for each subtask is: 1. Rainbow is successfully added; 2. There are no sunflowers in the field, suggest changing the tool to `GroundingDINOInpainting`; 3. There is no wooden barn in the image.

3.2.2 Quality competitor

In order to achieve results of higher quality and robustness, we leverage two generator agents to generate results and engage a quality Competitor agent $\mathcal{QC}$ to choose the superior one. For each edited image O, the competitor agent should compare their corresponding feedback F, and select the one that best aligns with the user's request.

In addition to competing in the generated results of these two agents, competition can also occur across different rounds. Specifically, the agent maintains a memory bank $\mathcal{M}$ to update the current best result $\mathcal{I}_{\text{best}}$. This process can be formalized as:

$$\mathcal{I}_{\text{best}}, F_{\text{best}} = \mathcal{QC}(\{O, F\}, \mathcal{M}), \tag{5}$$

$$\mathcal{M} = \{\mathcal{M}, (\mathcal{I}_{\text{best}}, F_{\text{best}})\}. \tag{6}$$

In this formula, $\mathcal{QC}(\cdot)$ represents the quality competitor function that determines the best result. $\{O, F\}$ is the set of different generator agents' output and feedback at the current run.

The competitor agent also plays a crucial role in deciding when to terminate the process. In each round, the agent checks if the current best result sufficiently meets the user's requirements. It is not necessary to proceed through all the rounds if the image quality is already deemed satisfactory. As discussed in Section 3.2.2, the Quality Competitor plays an important role in obtaining the best result and facilitating early termination.

3.3 Collaborative competitive agents

Our whole system contains two generative agents and one discriminator agent. The discriminator agent selects the best result for the user and provides feedback to the generative agents. Generative agents refine their strategies using both self-feedback and insights from peers. For example, if the first agent's plan is to "Perform subtask A first and then subtask B", and the second plan is to reverse the order of subtasks. If the discriminator agent recognizes that the second one gets a better result, the first agent may learn from this feedback to change the order for better performance. This demonstrates agents' cooperation via shared feedback. In Section 4.3.2, we demonstrate that such collaboration will make the system more robust and achieve better results.

In addition to cooperation, the discriminator will also promote competition between the two generative agents. During the initial stage of plan generation, different agents generate various outcomes due to randomness. These variations result in distinct edited results and subsequent feedback. Agents that produce poor results use feedback to improve their results, trying to produce better results than their counterparts. The discriminator agent benefits from this competitive mechanism as well. By learning from the edited images and feedback generated by various agents, the discriminator agent can provide more refined feedback and suggestions to the generator agent. If there's no discernible difference between the edited results, the discriminator agent will suggest selecting an alternative, more suitable tool to accomplish the sub-goal. The whole algorithm of Collaboration Competition Agent is shown in Algorithm 1.

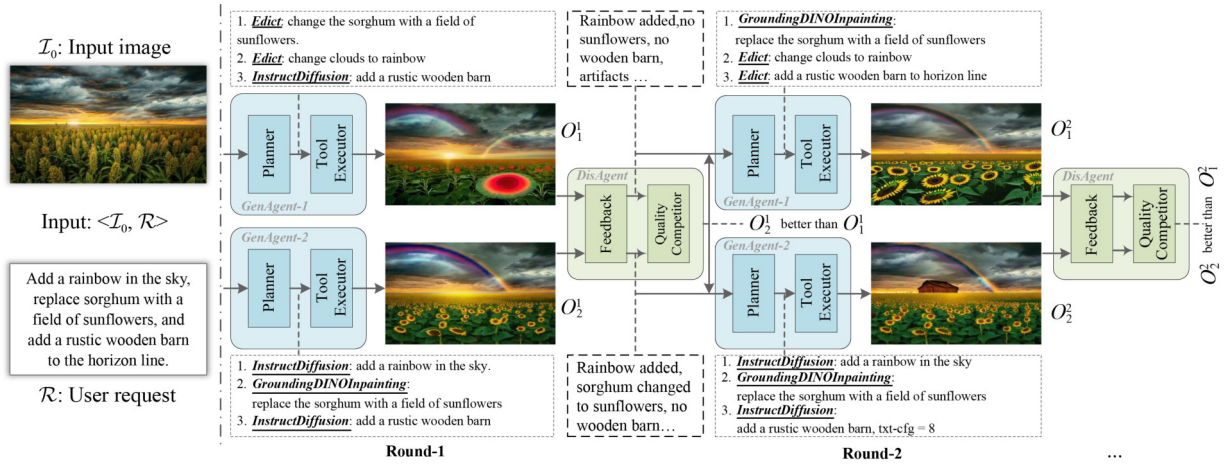


Fig. 1 The framework of our collaborative competitive agents system. Through providing feedback, the discriminator agent encourages the generator agent to engage in both collaborative learning and competition. The system's performance undergoes iterative optimization to effectively meet user requirements

Algorithm 1 Algorithm for CCA framework

Input: User request R, tool set T, input image $\mathcal{I}_0$, number of rounds M
Output: Edited image $\mathcal{I}_{\text{edited}}$
Other Symbols: Memory bank $\mathcal{M}$, Feedback F, feedback of each subtask $\{f_j\}$, subtasks of the plan $\{s_j\}$, round index m, intermediate edited results $\{o_j\}$, generator output O, Tool executor $\mathcal{E}$, Planner $\mathcal{P}$, Quality Competitor $\mathcal{QC}$, color blue denotes corresponding item from another generator agent.

function AGENT(:)
  $\{s_j\}^m \leftarrow \mathcal{P}(R, T, \{s_j\}^{m-1}, F^{m-1}, F^{m-1})$ ▷ get plan
  $o_1 \leftarrow \mathcal{I}_0$
  for all s_j in $\{s_j\}^m$ **do** $o_{j+1} \leftarrow \mathcal{E}(s_j, T(t_i), o_j, f_j)$ ▷ execute the plan
  $O^m \leftarrow o_{n+1}$
  $F^m \leftarrow \mathcal{FB}(O^m, R)$ ▷ Get the feedback
  return O^m, F^m

$\mathcal{M}, F, \{f_j\}^0, \{s_j\}^0 \leftarrow \emptyset$
for $m = 1$ to M **do**
  $O^m, F^m \leftarrow Agent1(R, T, \mathcal{I}_0)$; $O^m, F^m \leftarrow Agent2(R, T, \mathcal{I}_0)$
  $\{f_j\}, \{f_j\} \leftarrow F^m, F^m$ ▷ Decompose feedback
  $\mathcal{I}_{\text{best}}, F_{\text{best}} = \mathcal{QC}((O^m, F^m), (O^m, F^m), \mathcal{M})$ ▷ Quality compete
  $M \leftarrow \{M, (\mathcal{I}_{\text{best}}, F_{\text{best}})\}$ ▷ Update the memory
  if R is met by $\mathcal{I}_{\text{best}}$ **then break**
$\mathcal{I}_{\text{edited}} \leftarrow \mathcal{I}_{\text{best}}$

3.4 Hierarchical tool configuration

It's a huge challenge for agents, especially generator agents, to understand and accurately utilize various tools. Thus we propose a hierarchical tool configuration. For each tool, it should comprise a tool name, a description, and a user manual. The description succinctly articulates the tool's functionality, typically in one or several sentences. The manual offers detailed usage instructions, parameter effects, and input/output specifications.

Given the tool diversity, manuals are detailed. The Planner agent only reads the description of all the tools, decompose user request into sub-goals, and choose an appropriate tool for each sub-goal. The Tool Executor agent takes the user manual of the corresponding tool as input and designs the necessary parameters to use the tool.

4 Experiments

In this section, we initially explore the challenges inherent in the construction of such a system, providing an in-depth analysis of several key components in the process. Subsequently, we perform an ablation study to evaluate the efficacy of our individual components. Lastly, we compare our method with other image editing techniques to highlight the advantages of our approach.

4.1 Implementation details

We aim to build an automatic system to complete task A. The two most important parts are what kind of "brain" is used to think about the problem, and what tools are used to complete the task. For the Planner and Feedback part, we leverage GPT-4 to develop plans, generate feedback and suggestions. For the Tool Executor and Quality Competitor, we adopt GPT-3.5-turbo for its speed.

The type and quality of the toolset directly determine the complexity of tasks that can be accomplished and the quality of their completion. We furnish our generator agents with a diverse set of 20 tools, which fall broadly into several categories: Image Preprocessing, Localization, Understanding, Conditional Generation, and General Editing. For the discriminator agent, we deploy the state-of-the-art, open-source, large multi-modal model (LMM), LLaVA-1.5 [68], which is designed to understand and evaluate the quality of the edited images. LLaVA-1.5 excels in detailed captioning and VQA. Additionally, we also utilize an Aesthetic Predictor [69] to evaluate the overall perceptual quality of the results. Further details are provided in the supplementary material.

4.2 Step-by-step to build the framework

4.2.1 Yes/No questions rather than what

Feedback plays a pivotal role in enhancing the quality of editing. To glean information from the edited image, we design several questions and employ LLaVA [65] to answer these questions based on the edited image. Question design is key to eliciting precise feedback.

Initially, we attempted to ask questions like "What about the results generated in the figure?". However, we found that it was challenging for LLaVA to assess the extent of the edited item, and it was also difficult to generate a suggestion according to the vague answer. Figure 2 showcases that ambiguous questions may lead to confusion about whether the object has been successfully edited. In contrast, "Yes/No" questions tend to be answered with greater accuracy. Consequently, we modified our approach to pose "Yes/No" questions according to the user requirements, which yielded more precise feedback and better suggestions.

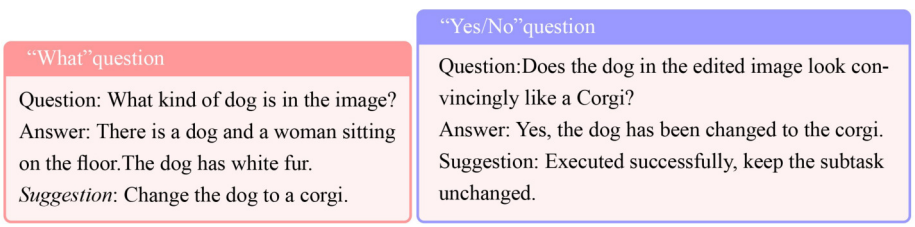


Fig. 2 The example user requirements is: "Change the dog to corgi and transform the image to pixel style". Yes/No questions can achieve more effective feedback

4.2.2 Tool diversity

While several studies [23,41] claim the ability to manage diverse types of editing tasks, such as object addition, removal, and replacement, these capabilities fall short of addressing the varied needs of practical applications. Conversely, even when multiple tools are employed for the same task, each may possess its own unique strength, potentially leading to synergistic enhancements. To illustrate this, we choose InstructDiffusion [41] as our baseline method, which can handle a wide range of image editing tasks following user instructions through the diffusion process. Meanwhile, we also incorporate EDICT [65] and GroundingDINO+Inpainting [19,70] to expand our toolset. As depicted in Fig. 3, the exclusive reliance on a single tool yields subpar result. In this example, the objective is to transform the house into a wooden one and replace the front stone with flowers. The single tool with InstructDiffusion failed to locate the front stone while with GroundingDINO's detection ability, multi-tool setting achieved a more reasonable result.

4.2.3 Stopping criteria

In most practical scenarios, we have observed that setting the maximum round, M, to 5 is sufficient to meet user requirements. Yet, task complexity can greatly change the needed round count. From a resource conservation standpoint, it is crucial for the quality competitor to determine the appropriate moment to terminate the process. We benchmarked this with 20 user requirements, tracking the tool calls by generator agents.

We observed that without the implementation of stopping criteria, the generator agent necessitated an average of 20 tool calls over 5 rounds. However, when we employed the quality competitor's judgment to determine early stoppage, the average number of tool calls reduced to 12 over approximately 3 rounds. Interestingly, we also found that the additional rounds and tool calls did not significantly enhance performance. This might be due to the feedback at this stage not providing explicit guidance on improving the results.

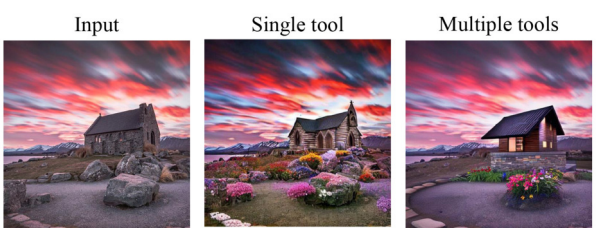


Fig. 3 Comparison of single tool vs. multiple tools. Prompt: Replace the house with a wooden one, and turn the stones in front of the house into flowers

4.3 Ablation study

4.3.1 Coarse-to-fine tool usage

Given that current large language models such as GPT-3.5-turbo/GPT-4 [2,3] possess a finite context length, it presents a challenge for the planner to directly select proper parameters for calling tools while formulating the plan. We tackle this with a coarse-to-fine tool usage strategy. The planner inputs tool descriptions and selects tools, while the executor interprets their specific instructions. We find that a one-step plan carries an over 20% risk of producing an incorrect format. However, this risk diminishes to less than 10% when employing the hierarchical setting.

Taking the user request, "Enrich wooden frames to the photo and adjust the longer side to 512" as an example, we compare different tool usages in Fig. 4. Analysis reveals both plans appear fitting, but InstructDiffusion in the first can't manage size changes directly. In comparison, the hierarchical tool usage breaks the task into subtasks for a logical plan. This highlights the first approach's inadequate understanding of tool usage.

Furthermore, we carried out an experiment to verify whether Tool Executor can adjust parameters under the hierarchical tool usage. For a clearer observation, the toolset is restricted to a single tool, `InstructDiffusion`. We observed that as the number of rounds increases, the Tool Executor gradually increases one of the key parameters txt-cfg (text classifier-free guidance), as depicted in Fig. 5. It's apparent that the initial value (txt-cfg = 4) does not fulfill the requirements and it progressively grows to txt-cfg = 8. This observation further underscores the pivotal role that feedback can play in hierarchical tool usage.

4.3.2 Effect of collaboration and competition

Our system relies on collaboration for efficient, high-quality planning, and competition to enhance generated results and boost system robustness. We tested this across four settings in 100 instances. a) Our full model devoid of both collaboration and competition, b) our model excluding collaboration, c) our model excluding competition, and d) our full model. We compare the first three settings against the final setting, prompting users to determine which results they perceived as superior. The results are demonstrated in Fig. 6. Both strategies contribute to improved outcomes, with the best result achieved by incorporating both elements.

4.4 Comparison

We benchmark our CCA against techniques like InstructPix2Pix [23], MagicBrush [54], InstructDiffuion [41],

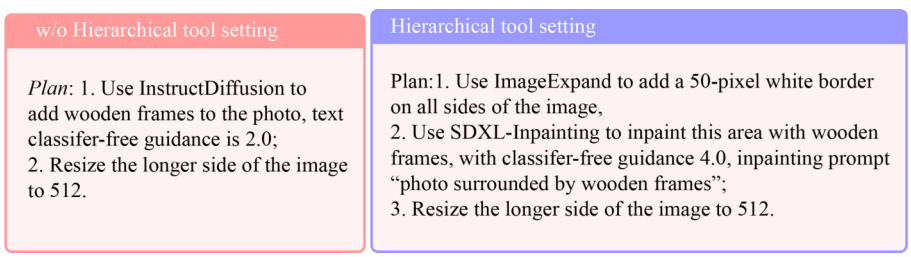


Fig. 4 The example of the effect of the hierarchical tool setting. The given user request is "Enrich wooden frames to the photo and adjust the longer side to 512"

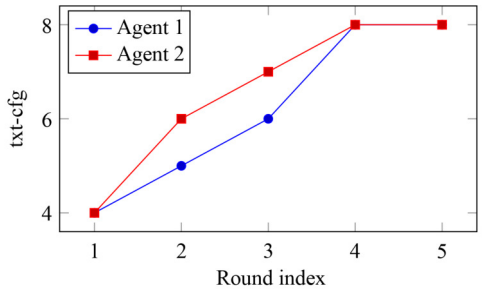


Fig. 5 The evolution of classifier-free guidance as the round index increases

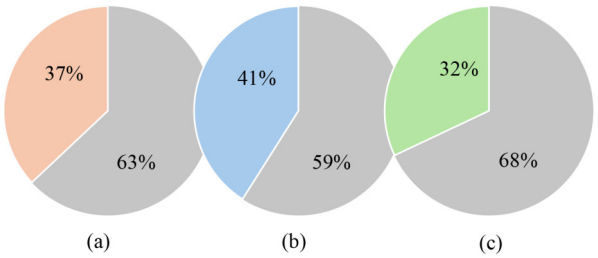


Fig. 6 Ablation studies of our method by removing collaboration and competition individually, and both together. The grey segment indicates the proportion of users favoring our approach. (a) w/o collaboration; (b) w/o competition; (c) w/o both

and VisProg [67]. The comparison results, as depicted in Fig. 7, indicate that all previous works struggle to manage such complex cases. Despite VisProg's ability to dissect intricate tasks, it fails to execute edits in alignment with user requirements. This underscores the value of using agents cooperatively and competitively. Additionally, we also perform a quantitative comparison based on human preferences, with the results included in the supplementary material.

4.5 Discussion

Our goal is to propose a universal agent-based framework to address complex user requirements rather than focusing on a particular task. Image editing is our first significant achievement on the path toward this goal. The CCA framework can also be applied to *other tasks* such as text-to-image generation. We leave this application in the supplementary material. Within our proposed CCA framework, GPT-4 plays a significant role in planning and reflection. However, our method also performs well when utilizing GPT-3.5 Turbo as an alternative. It demonstrate that our method is robust to the choice of LLMs. Related results will also be included in the supplementary material.

5 Conclusion

This paper introduces a novel generative framework, Collaborative Competitive Agents (CCA), that employs multiple LLM-based agents to tackle intricate image editing challenges in practice. The key strength of the CCA system lies in its capacity to decompose complex tasks using multiple agents, resulting in a transparent generation process. This enables agents to learn from each other, fostering collaboration and competition to fulfill user requirements. The study's primary contributions entail the proposition of a new multi-agent-based generative model, an examination of the relationships between multiple agents, and extensive experimentation in the area of image editing. This work represents a step forward in AI research, with the potential to influence various fields.

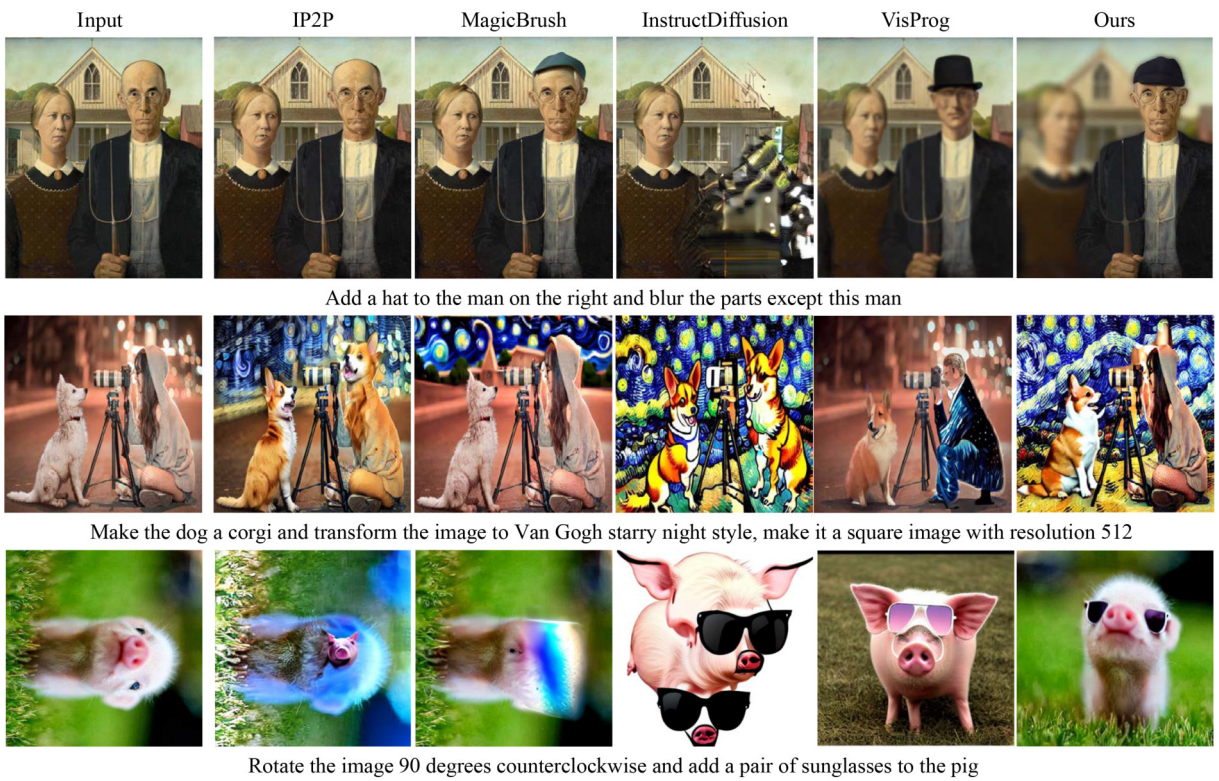


Fig. 7 Qualitative comparison between InstructPix2Pix [23], MagicBrush [54], InstructDiffusion [41], VisProg [67], and ours

Competing interests The authors declare that they have no competing interests or financial conflicts to disclose.

Appendixes

Comparison with previous methods

In this section, we conduct a comprehensive comparison of our approach with other methods, encompassing both quantitative results and qualitative results.

Human preference study

We showcase a human preference study that compares our method with previous state-of-the-art techniques, as demonstrated in Table A1. VisProg [67] is not included in this table because it is specific design for straightforward object removal tasks and certain low-level operations, typically falls short when subjected to a broader comparison.

We have constructed an interface using Gradio to facilitate this comparison of user preferences. The interface presents the input image, editing instruction, and outcomes from various methods to the user. In addition, we ask human participants to select both the result that best adheres to the instruction and the one that exhibits the highest quality, respectively. The table reveals that 47 percent of respondents concur that our method, CCA, more effectively fulfills user requirements, while 44 percent believe that our results showcase superior quality.

Effect of number of agents

We add some quantitative results in Table A2, and we will add related results in revised version. For CLIP score, system with 2 agents performs better than 1 agent by 1.0 and 3 agents outperforms 2 agents by 0.04. For aesthetic score, 2-agent system performs better than its counterparts. L1 distance measures the distance between input image and edited image. 2-agent system achieves good CLIP score and aesthetic score, and also maintains the distance between input and edited image.

More qualitative results

We show more qualitative results in Figs. A1, A2, and A3. From the first two figures, we compare with previous methods on more cases. The third figure aims to further underscore the efficacy of our approach. Contrary to conventional image editing techniques, our agent-based system exhibits superior performance in tasks such as background removal, precise black and white conversion of specific regions, and sticker addition.

Prompt templates

The prompt supplied to the LLMs can substantially influence its performance. Hence, We introduce the main prompt templates employed within our framework. The necessary input variables are highlighted in italics and enclosed by <>.

Table A1 Human preference. In this table, IP2P, MB, and ID are abbreviations for InstructPix2Pix [23], MagicBrush [54], and InstructDiffusion [41], respectively

	IP2P	MB	ID	CCA
Txt-alignment	11%	21%	21%	47%
Visual quality	16%	17%	23%	44%

Table A2 Comparison of different methods

CLIP score	
Instruct diffusion	31.21
1 Agents	31.09
2 Agents	32.09
3 Agents	32.13
Aesthetic score	
Instruct diffusion	6.16
1 Agents	6.20
2 Agents	6.31
3 Agents	6.07
L1 distance	
Instruct diffusion	134.25
1 Agents	130.42
2 Agents	131.22
3 Agents	138.66

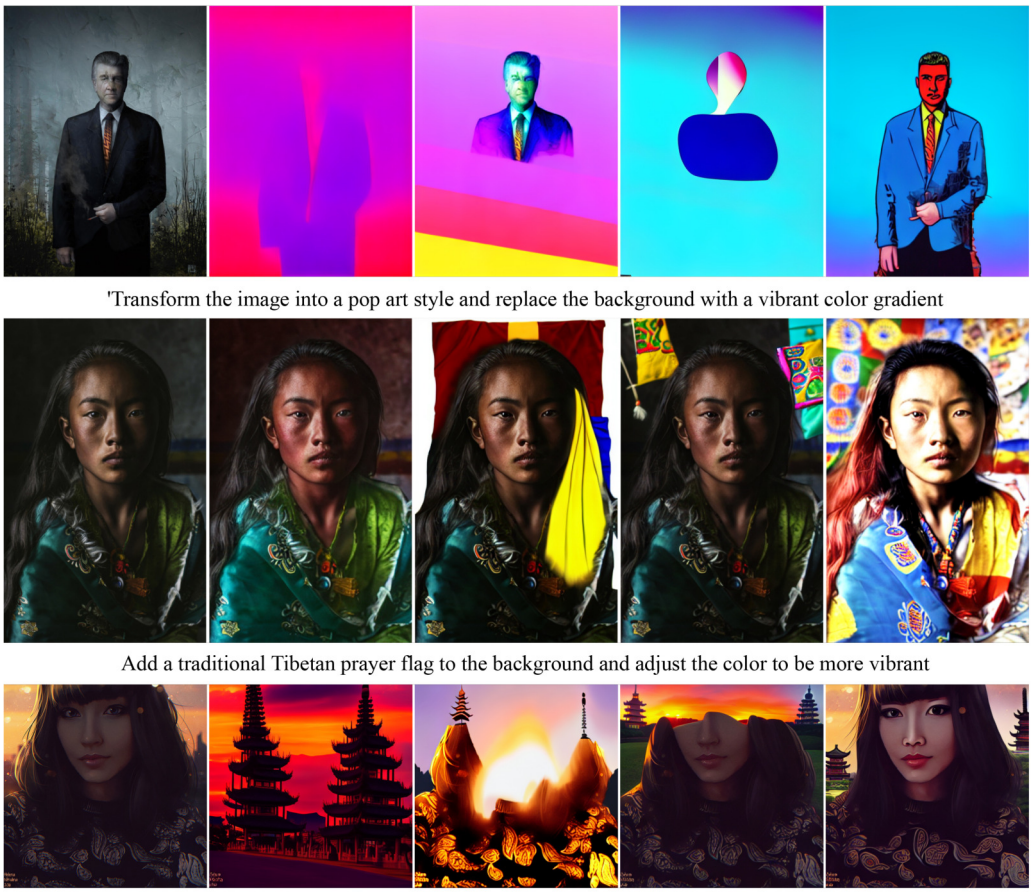
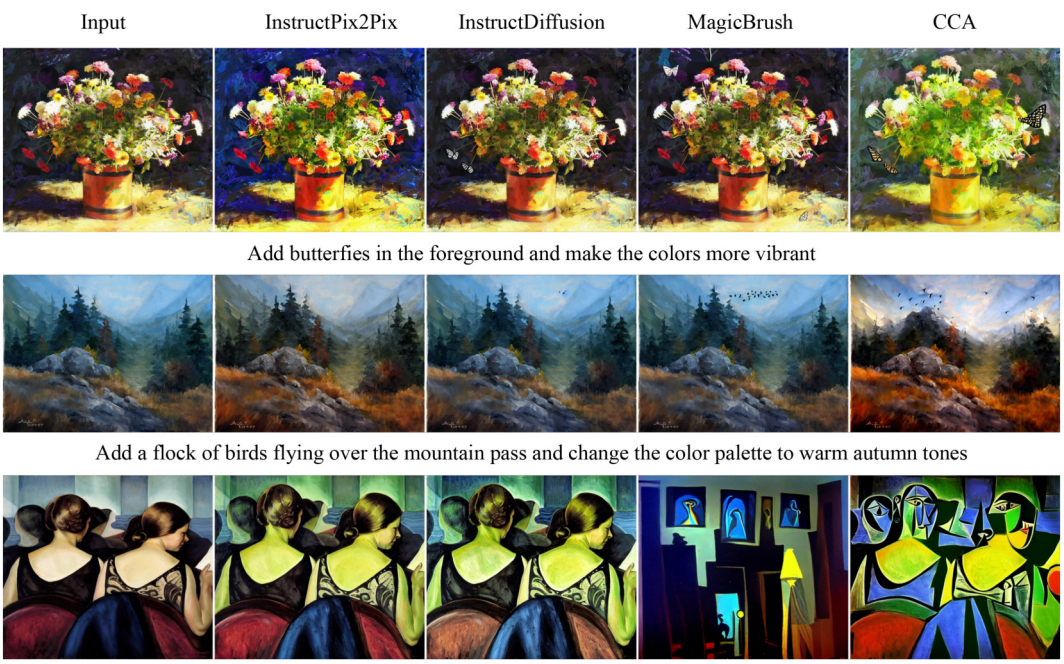


Fig. A1 Comparison results between previous methods and our CCA. From left to right: input image, results from InstructPix2Pix [23], MagicBrush [54], InstructDiffusion [41], and our proposed CCA. Below the images is the editing instruction



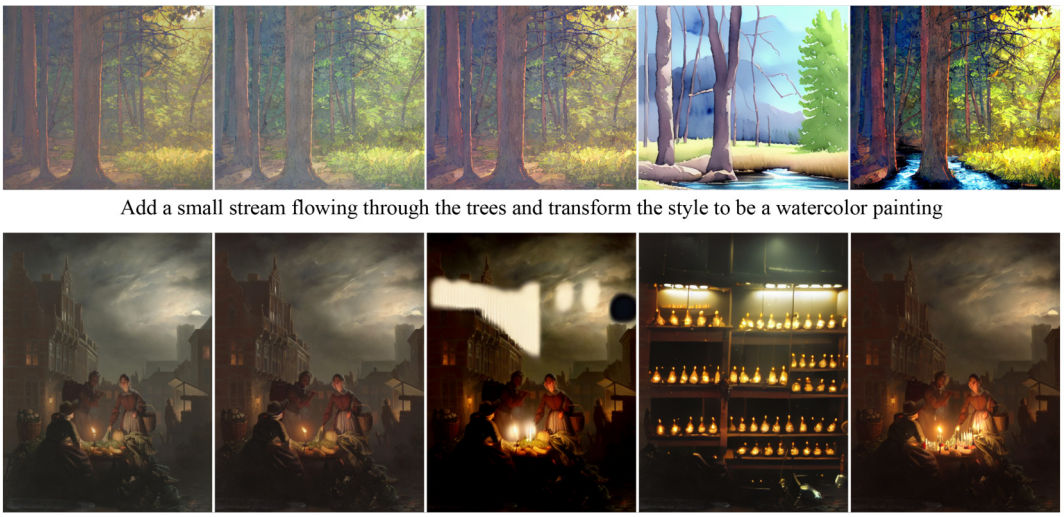


Fig. A2 Comparison results between previous methods and our CCA. From left to right: input image, results from InstructPix2Pix [23], MagicBrush [54], InstructDiffusion [41], and our proposed CCA. Below the images is the editing instruction

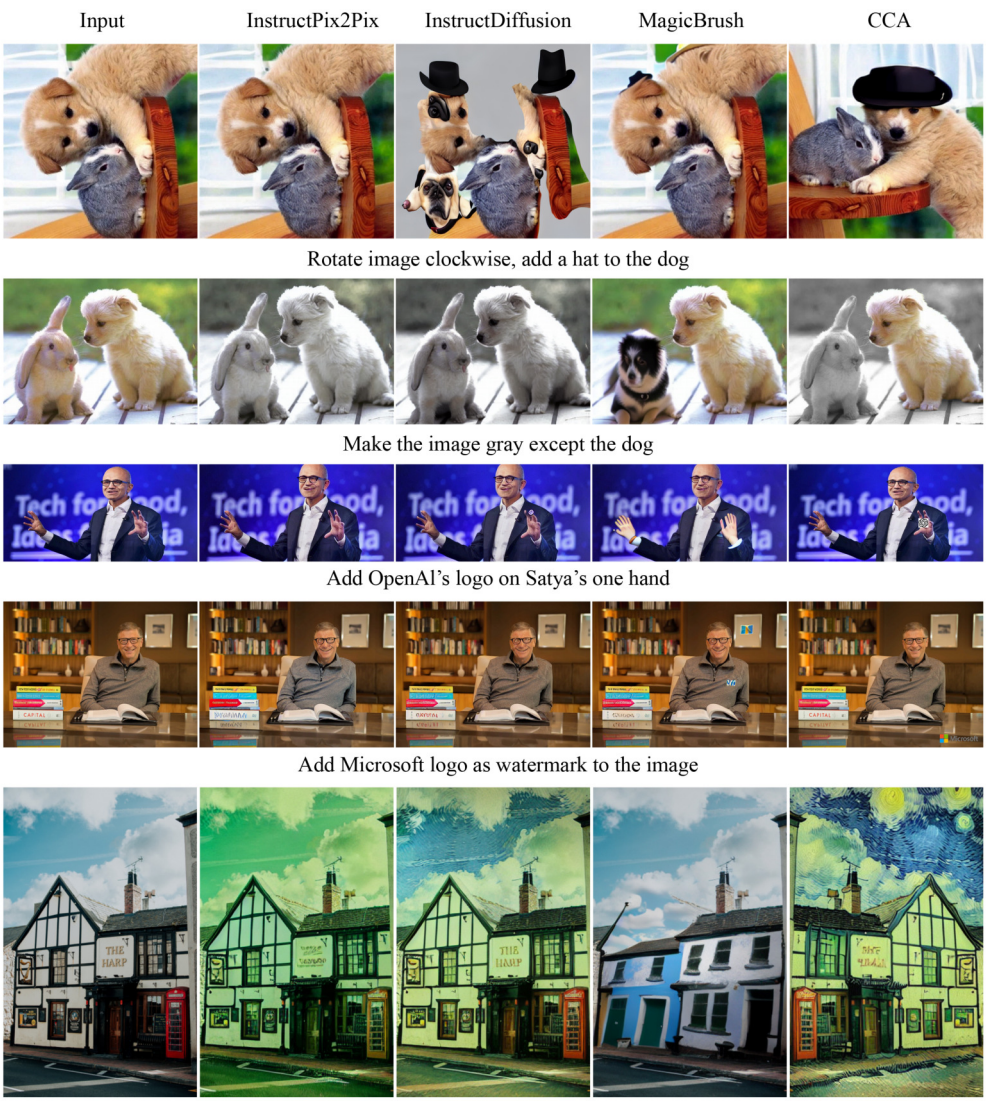


Fig. A3 Comparison results between previous methods and our CCA. From left to right: input image, results from InstructPix2Pix [23], MagicBrush [54], InstructDiffusion [41], and our proposed CCA. Below the images is the editing instruction

Further prompts, including tool descriptions and the user manual, will be made available concurrent with the code release soon.

Planner agent $\mathcal{P}$ (first round of planning)

I want to edit the image <*IMAGE PATH* >using user's instruction '<*EDITING REQUEST* >'. Please help me decompose this task into several subtasks. All images should be saved at the same folder 'image/'.

Each subtask should be short and specific that can be done by only a single tool. Each subtask should only be tried once. Each subtask should be described in a single line.
The tool should be one of the following:
‘‘‘
<*TOOL NAMES* >
‘‘‘

The detailed description of each tool is as follows:
‘‘‘
<*TOOL DESCRIPTIONS* >
‘‘‘

For example, if the user requirement is 'Create a vintage-style portrait of a person with a hat and adjust the image to have a sepia tone, with the longest side being 800 pixels.' with an given input image, it can be decomposed into the following steps:
1. Resize the image to have its longest side at 800 pixels using `Resize`; 2. Add a vintage-style hat to the person in the image using `Instructdiffusion`; 3. Apply a sepia tone filter to the entire image `Edict`.

Do not include specific input/output image path in subtasks. If you have to resize the image, put it at the first. The final response should be concise and clear.

Now give me the plan.

Planner agent $\mathcal{P}$ (reflection)

I want to edit the image <*IMAGE PATH* >using '<*EDITING REQUEST* >'. Please help me decompose this task into several subtasks. All images should be saved at the same folder 'image/'

Each subtask should be short and specific that can be done by only a single tool. Each subtask should only be tried once. Each subtask should be described in a single line.
The tool must be one of the following:
‘‘‘
<*TOOL NAMES* >
‘‘‘

The detailed description of each tool is as follows:
‘‘‘
<*TOOL DESCRIPTIONS* >
‘‘‘

Currently, I have decomposed it into the following plans (subtask with related tool):
‘‘‘
<*SUBTASKS* >
‘‘‘

The feedback obtained by executing step by step is:
‘‘‘
<*FEEDBACK* >
‘‘‘
Besides, we also have another plan:
‘‘‘
<*PLAN* >
‘‘‘
This plan obtained the following feedback:
‘‘‘
<*FEEDBACK* >
‘‘‘
Comparing the current plan and the referenced plan, Do you think I should change the order of the subtasks or modify the content of subtasks? If yes, please tell me the new plan to improve the editing quality. You should choose one specific tool for each subtask. If you think I should only change the tool or the input to the tool, please respond with "No".

Feedback agent $\mathcal{FB}$ (question generation)

We want to edit the image using instruction '<*EDITING REQUEST* >'. The detailed description of the input image is '<*CAPTION* >'. Suppose we have edited the image, please design some questions to ask human to judge the quality of the edited image.

For example, if the task is 'Transform a daytime cityscape photo into a nighttime scene with lit streetlights and a full moon, with the long side being 800 pixels.'. The description of the image is 'A cityscape photo with a busy street, tall buildings, and people walking around.' Then the questions can be like: 1. Is the overall setting of the picture a nighttime scene? \n 2.Are streetlights visible in the original image?\n 3.Is the moon present in the photo? \n 4. Is the size

of image changed to 800?

The questions should be concrete and clear and do not include hallucination. **Yes or No questions are preferred**. Reduce overlap between the questions. The questions should be composed of local and global editing effects. You need to ensure that parts unrelated to the editing requirements remain unchanged.\n The number of questions should be no more than five. Each question should be concise and clear.

Now give me the questions.

Tool set

Agents employ tools to accomplish specific tasks. Our goal is to build an automated pipeline for image editing where tools serve as critical components. For customized editing tasks, we first preprocess the image, followed by the selection of suitable tools. We then configure the associated parameters and ultimately obtain the results. We provide the following tools for the agent:

- **Resize** This function alters the dimensions of an image to a specified resolution. It accepts the image and the desired resolution as inputs and produces a resized image as the output.
- **Paste** This feature allows for the insertion of a smaller image onto a specified position within a larger one. It requires the base image, the secondary image, and the position coordinates for pasting as inputs, resulting in a composite image as the output.
- **Blending** Merge two images at a specific area of the base image. It takes the base image, the image to blend, the position coordinates where blending should start, and a blending strength parameter as input. This function outputs the base image with the second image blended into the specified area.
- **InstructDiffusion** [41] A text-guided image editing tool that accepts an image and an editing prompt as inputs and generates the edited image as output.
- **LLaVA** [68] Give the detailed caption of the image and answer questions about the image. It takes the image and question as input and outputs the answer to the question.
- **AestheticScore** [69] Used to assess the aesthetic quality of the image. It takes the image as input and outputs the aesthetic score.
- **ImageDifferenceLLaVA** Based on LLaVA 1.5 [68], this tool concat the input image and edited image and tells the difference between two parts. It takes the original image and the edited image as input, and returns the string describes the differences between them.
- **GroundingDINO** [70] Utilizing detection prompt, this tool identifies and segments specific elements within an image. It requires an image and a corresponding detection prompt as inputs and outputs the mask.
- **Prompt2Prompt** [24] Designed for text-guided image editing, this tool adeptly handles tasks such as object replacement by taking an original image, a base prompt, and a target prompt as inputs to produce the edited image.
- **Crop** This tool is designed to selectively crop a specified region of an image, requiring the image and the target region's coordinates as inputs to deliver the cropped output.
- **RGB2Gray** This tool converts RGB color images into grayscale, accepting a color image as input and producing a single-channel grayscale image as output.
- **GaussianBlur** This function applies a Gaussian blur to the input image. It requires the image and the Gaussian kernel size to return the blurred image.
- **RotateClockwise** Rotate the image in the clockwise direction. It takes the image as input and outputs the rotated image.
- **RotateCounterClockwise** Rotate the image in the counterclockwise direction. It takes the image as input and outputs the rotated image.
- **EnhanceColor** It is used to adjust the color balance of an image. It takes the image and an enhancement factor as input and outputs the enhanced image.
- **FlipHorizontal** This operation mirrors the image along the horizontal axis. It accepts an image as input and outputs the horizontally flipped version.
- **AddLogo** This utility embeds a logo onto an image at a specified location. It requires the original image, the logo file, and the coordinates for the logo's placement as inputs to output an image with the logo superimposed.
- **AddWatermark** This function overlays a watermark onto an image, specifically positioning it in the bottom right corner. It takes the original image and a watermark image as inputs, with an optional alpha parameter to adjust the watermark transparency. The watermark is blended over the original image, ensuring visibility while maintaining the integrity of the image content.
- **Inpainting** This tool performs inpainting on a specified masking area using SDXL-Inpainting. It takes the image, mask, and target prompt as inputs, and outputs the image whose masked region is inpainted to the target prompt.
- **GetSize** This tool gets the size of the image. It takes the image as input and outputs the size of the image.

Application of other tasks

We chose the image editing task for our paper because it is relatively easy to obtain feedback from the generated results. Besides, it necessitates the capability to analyze complex instructions for planning, execution, and feedback integration, which highlights the advantages of our framework more effectively. We have also explored the text-to-image task and present some preliminary results in Fig. A4. When given a prompt, our CCA can produce images that align more closely with the prompt than when a tool is used directly. This advantage is primarily due to the feedback and iterative

Fig. A4 Additional application of CCA on text-to-image task. For each case, the left is generated by Stable Diffusion 1.5 and the right is generated by our framework

optimization mechanisms within the CCA system. We will include more details in the final version of this document. These tasks suggest the practicality of our method. We are optimistic about the general applicability of our approach and intend to explore its potential in a broader array of tasks, including robotics and multi-agent systems, in future research.

Running time analysis

We obtained the processing times for different configurations of Large Language Models (LLMs), notably including GPT-3.5-Turbo and GPT-4-Turbo. Twelve experimental trials were conducted to ascertain the duration of each. After discarding the outliers —the most and the least time-consuming trials—we computed the mean processing time. The averaged duration for the GPT-3.5-Turbo amounted to 4.6 minutes, whereas for the GPT-4-Turbo, it requires about 11.5 minutes.

Limitations

While the CCA attains commendable results, our experiments reveal two primary limitations. First, the entire pipeline processes the subtasks in a sequential manner. This means that each task relies solely on the output of its preceding subtask, without access to earlier images, which could potentially lead to error accumulation. Second, agents may lack a comprehensive understanding of the tool, necessitating multiple rounds of exploration to achieve satisfactory results. In future work, we plan to introduce a memory module to agents so that they can deepen the understanding of the tool during the reflection process.

Broader impact

The novel generative model introduced in this paper, the Collaborative Competitive Agents (CCA), leverages multiple Large Language Models (LLMs) to perform complex tasks and demonstrates a significant advancement in image editing. While our work primarily aims to enhance the capabilities of generative models, the broader implications of this research must be carefully considered, including the potential long-term impacts on society, the environment.

Postive impact:

- Innovation in creative industries: The CCA model can significantly improve the efficiency and quality of content creation in art, entertainment, and design, leading to new products and services.
- Enhanced decision making: By decomposing complex tasks, the CCA system can aid in areas requiring intricate analysis, such as climate modeling, urban planning, and logistics. Accessibility in Technology: The system's ability to interpret and execute detailed instructions can make sophisticated technological tools more accessible to a broader audience, democratizing design and creativity.

Negative impact:

- Misuse of synthetic media: The ability of the CCA to create realistic and complex images raises concerns about the production of synthetic media for disinformation, identity theft, or other malicious intents.
- Bias propagation: If the LLMs or datasets used in the CCA inherit societal biases, the generated content might perpetuate or exacerbate these biases, leading to unfair or discriminatory outcomes.

The CCA model presents a powerful tool for generative tasks, with the potential to contribute positively to various sectors. However, it is crucial to anticipate the ethical and societal challenges posed by such advancements. It is the responsibility of the research community to engage in ongoing discussions about AI ethics and to develop strategies that mitigate risks and ensure the responsible use of AI technologies.

Success rate of various open-source models

For each planned task, the agent views all tool names and descriptions and selects the appropriate tool. Tool name is extracted from the agent's response. Given the selected tool name and plan, the agent views the detailed cookbook for the tool and provides the input arguments. For example, if the tool is tool-1, the related arguments are arg1, arg2, and arg3. We require the agent to respond in a specific format, e.g., tool-1 @@ arg1 ↔ arg2 ↔ arg3. This structured format facilitates the extraction of arguments and the subsequent execution of the tool. The hierarchical design enhances the success rate of formatting tool execution.

We evaluate the success rate of various models in the following Table A3. We assessed 100 editing prompts to determine whether the LLM could follow our instructions to generate the formatted plan and subsequently format the appropriate tool usage for each sub-plan (tool-1 @@ arg1 ↔ arg2 ↔ arg3).

Additional visual results of various open-source models

We also provided several visual examples to demonstrate the comparative performance of different open-source models on image editing tasks in our framework. As shown in Fig. A5,

Table A3 Model success rate

Model	Success rate
Mistral-7B-Instruct-v0.1	0.18
Mistral-7B-Instruct-v0.2	0.74
Qwen2-7B-Instruct	0.97
DeepSeek-V2	1.0

Fig. A5 Visual comparison of image editing capabilities across different large language models. From left to right: Input image, Mistral, Qwen, DeepSeek, and ChatGPT results given various editing prompts

we tested various editing scenarios to evaluate the models' capabilities. The results reveal notable differences in performance: Mistral-7B shows basic editing capabilities but often struggles with complex modifications and maintaining image coherence, particularly in cases requiring sophisticated semantic understanding or fine-grained adjustments. Qwen2-7B-Instruct demonstrates significantly better performance, with superior handling of complex editing tasks, particularly in maintaining context consistency and generating realistic modifications. The visual results show improved detail preservation and more natural transitions in edited regions. DeepSeek-V2 produces the most sophisticated editing results among the open-source models. The visual examples highlight its capability to handle nuanced modifications while preserving image quality and semantic coherence, especially in challenging scenarios involving multiple objects or complex transformations. ChatGPT's results are included as a commercial baseline for comparison, showing comparable performance to DeepSeek-V2 in most cases, though with notably higher computational requirements and associated costs. These visual comparisons demonstrate the rapid progress in open-source models' capabilities, particularly in image editing applications. The qualitative improvements in newer model versions suggest promising directions for future development in this domain.

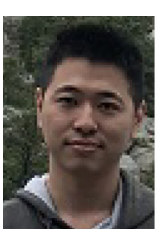
References

1. OpenAI. Gpt-4v(ision) system card. See api.semanticscholar.org/CorpusID:263218031} website, 2023
2. OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, et al. Gpt-4 technical report. 2023, arXiv preprint arXiv: 2303.08774
3. Ouyang L, Wu J, Jiang X, Almeida D, Wainwright C L, Mishkin P, Zhang C, Agarwal S, Slama K, Ray A, Schulman J, Hilton J, Kelton F, Miller L, Simens M, Askell A, Welinder P, Christiiano P, Leike J, Lowe R. Training language models to follow instructions with human feedback. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 2011
4. 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. 2023, arXiv preprint arXiv: 2302.13971
5. Touvron H, Martin L, Stone K, Albert P, Almahairi A, et al. Llama 2: open foundation and fine-tuned chat models. 2023, arXiv preprint arXiv: 2307.09288
6. Yao S, Chen H, Yang J, Narasimhan K. WebShop: towards scalable real-world web interaction with grounded language agents. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 1508
7. Qian C, Cong X, Liu W, Yang C, Chen W, Su Y, Dang Y, Li J, Xu J, Li D, Liu Z, Sun M. Communicative agents for software development. 2023, arXiv preprint arXiv:2307.07924
8. Swan M, Kido T, Roland E, dos Santos R P. Math agents:

computational infrastructure, mathematical embedding, and genomics. 2023, arXiv preprint arXiv: 2307.02502
9. Kalvakurthi V, Varde A S, Jenq J. Hey Dona! Can you help me with student course registration? 2023, arXiv preprint arXiv: 2303.13548
10. Park J S, O'Brien J, Cai C J, Morris M R, Liang P, Bernstein M S. Generative agents: interactive simulacra of human behavior. In: Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology. 2023, 2
11. Goodfellow I J, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. Generative adversarial nets. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. 2014, 2672–2680
12. Brock A, Donahue J, Simonyan K. Large scale GAN training for high fidelity natural image synthesis. In: Proceedings of the 7th International Conference on Learning Representations. 2019
13. Karras T, Laine S, Aila T. A style-based generator architecture for generative adversarial networks. In: Proceedings of 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, 4401–4410
14. Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and improving the image quality of StyleGAN. In: Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020
15. Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. 2020, 574
16. Song Y, Sohl-Dickstein J, Kingma D P, Kumar A, Ermon S, Poole B. Score-based generative modeling through stochastic differential equations. In: Proceedings of the 9th International Conference on Learning Representations. 2021
17. Dhariwal P, Nichol A. Diffusion models beat GANs on image synthesis. In: Proceedings of the 35th International Conference on Neural Information Processing Systems. 2021, 672
18. Karras T, Aittala M, Aila T, Laine S. Elucidating the design space of diffusion-based generative models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 1926
19. Podell D, English Z, Lacey K, Blattmann A, Dockhorn T, Müller J, Penna J, Rombach R. SDXL: improving latent diffusion models for high-resolution image synthesis. In: Proceedings of the 12th International Conference on Learning Representations. 2024
20. Nichol A Q, Dhariwal P. Improved denoising diffusion probabilistic models. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 8162–8171
21. Hang T, Gu S, Geng X, Guo B. Improved noise schedule for diffusion training. 2024, arXiv preprint arXiv: 2407.03297
22. Wang T, Yang Q, Wang R, Sun D, Li J, Chen Y, Hu Y, Yang C, Kimura T, Kara D, Abdelzaher T F. Fine-grained control of generative data augmentation in IoT sensing. In: Proceedings of the 38th Annual Conference on Neural Information Processing Systems. 2024
23. Brooks T, Holynski A, Efros A A. InstructPix2Pix: learning to follow image editing instructions. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 18392–18402
24. Hertz A, Mokady R, Tenenbaum J, Aberman K, Pritch Y, Cohen-Or D. Prompt-to-prompt image editing with cross-attention control. In: Proceedings of the 11th International Conference on Learning Representations. 2023
25. Meng C, He Y, Song Y, Song J, Wu J, Zhu J Y, Ermon S. SDEdit: guided image synthesis and editing with stochastic differential equations. In: Proceedings of the 10th International Conference on Learning Representations. 2022
26. Sutton R S, Barto A G. Reinforcement Learning: An Introduction. 2nd ed. Cambridge: MIT Press, 2018
27. Xi Z, Chen W, Guo X, He W, Ding Y, et al. The rise and potential of large language model based agents: a survey. Science China Information Sciences, 2025, 68(2): 121101
28. Weng L. LLM powered autonomous agents. See Lilianweng.github.io website, 2023
29. Deng Q, Yang Q, Yuan R, Huang Y, Wang Y, Liu X, Tian Z, Pan J, Zhang G, Lin H, Li Y, Ma Y, Fu J, Lin C, Benetos E, Wang W, Xia G, Xue W, Guo Y. ComposerX: multi-agent symbolic music composition with LLMs. In: Proceedings of the 25th International Society for Music Information Retrieval Conference. 2024
30. Schick T, Dwivedi-Yu J, Dessì R, Raileanu R, Lomeli M, Hambro E, Zettlemoyer L, Cancedda N, Scialom T. Toolformer: language models can teach themselves to use tools. In: Proceedings of the 36th Annual Conference on Neural Information Processing Systems. 2023
31. Wu Y, Yang X. A glance at in-context learning. Frontiers of Computer Science, 2024, 18(5): 185347
32. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, Chi E H, Le Q V, Zhou D. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 1800
33. Yao S, Zhao J, Yu D, Du N, Shafran I, Narasimhan K R, Cao Y. ReAct: synergizing reasoning and acting in language models. In: Proceedings of the 11th International Conference on Learning Representations. 2023
34. Madaan A, Tandon N, Gupta P, Hallinan S, Gao L, Wiegreffe S, Alon U, Dziri N, Prabhumoye S, Yang Y, Gupta S, Majumder B P, Hermann K, Welleck S, Yazdanbakhsh A, Clark P. SELF-REFINE: iterative refinement with self-feedback. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 2019
35. Yang Z, Wang J, Li L, Lin K, Lin C C, Liu Z, Wang L. *Idea2img*: iterative self-refinement with GPT-4V for automatic image design and generation. In: Proceedings of the 18th European Conference on Computer Vision. 2025
36. Shen Y, Song K, Tan X, Li D, Lu W, Zhuang Y. HuggingGPT: solving AI tasks with ChatGPT and its friends in hugging face. In: Proceedings of the 37th Annual Conference on Neural Information Processing Systems. 2023
37. Driess D, Xia F, Sajjadi M S M, Lynch C, Chowdhery A, Ichter B, Wahid A, Tompson J, Vuong Q, Yu T, Huang W, Chebotar Y, Sermanet P, Duckworth D, Levine S, Vanhoucke V, Hausman K, Toussaint M, Greff K, Zeng A, Mordatch I, Florence P. PaLM-E: an embodied multimodal language model. In: Proceedings of the 40th International Conference on Machine Learning. 2023, 340
38. Li G, Hammoud H A A K, Itani H, Khizbullin D, Ghanem B. CAMEL: communicative agents for "mind" exploration of large language model society. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. 2023, 2264
39. Chen W, Su Y, Zuo J, Yang C, Yuan C, Chan C M, Qin Y, Lu Y, Hung Y H, Qian C, Qin Y, Cong X, Xie R, Liu Z, Sun M, Zhou J. AgentVerse: facilitating multi-agent collaboration and exploring emergent behaviors. 2023, arXiv preprint arXiv: 2308.10848
40. Chan C M, Chen W, Su Y, Yu J, Xue W, Zhang S, Fu J, Liu Z. ChatEval: towards better LLM-based evaluators through multi-agent debate. In: Proceedings of the 12th International Conference on Learning Representations. 2024
41. Geng Z, Yang B, Hang T, Li C, Gu S, Zhang T, Bao J, Zhang Z, Li H, Hu H, Chen D, Guo B. InstructDiffusion: a generalist modeling interface for vision tasks. In: Proceedings of 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024
42. Hang T, Yang H, Liu B, Fu J, Geng X, Guo B. Language-guided face animation by recurrent styleGAN-based generator. IEEE Transactions on Multimedia, 2023, 25: 9216-9227
43. Mokady R, Hertz A, Aberman K, Pritch Y, Cohen-Or D. Null-text

inversion for editing real images using guided diffusion models. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 6038–6047
44. Johnson J, Alahi A, Fei-Fei L. Perceptual losses for real-time style transfer and super-resolution. In: Proceedings of the 14th European Conference on Computer Vision. 2016, 694–711
45. Gatys L A, Ecker A S, Bethge M. Image style transfer using convolutional neural networks. In: Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. 2016, 2414–2423
46. Gu S, Chen C, Liao J, Yuan L. Arbitrary style transfer with deep feature reshuffle. In: Proceedings of 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2018, 8222–8231
47. Ding Z, Li P, Yang Q, Li S, Gong Q. Regional style and color transfer. In: Proceedings of the 5th International Conference on Computer Vision, Image and Deep Learning. 2024, 593–597
48. Zhu J Y, Park T, Isola P, Efros A A. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: Proceedings of 2017 IEEE International Conference on Computer Vision. 2017, 2223–2232
49. Isola P, Zhu J Y, Zhou T, Efros A A. Image-to-image translation with conditional adversarial networks. In: Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition. 2017, 1125–1134
50. Bertalmio M, Sapiro G, Caselles V, Ballester C. Image inpainting. In: Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques. 2000, 417–424
51. Criminisi A, Perez P, Toyama K. Object removal by exemplar-based inpainting. In: Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2003
52. Sun J, Yuan L, Jia J, Shum H Y. Image completion with structure propagation. In: Proceedings of the ACM SIGGRAPH 2005 Papers. 2005, 861–868
53. Yang B, Gu S, Zhang B, Zhang T, Chen X, Sun X, Chen D, Wen F. Paint by example: exemplar-based image editing with diffusion models. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 18381–18391
54. Zhang K, Mo L, Chen W, Sun H, Su Y. MagicBrush: a manually annotated dataset for instruction-guided image editing. In: Proceedings of the 37th Annual Conference on Neural Information Processing Systems. 2023
55. Xia W, Zhang Y, Yang Y, Xue J H, Zhou B, Yang M H. GAN inversion: a survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2023, 45(3): 3121-3138
56. Shen Y, Gu J, Tang X, Zhou B. Interpreting the latent space of GANs for semantic face editing. In: Proceedings of 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020, 9243–9252
57. Zhu J, Shen Y, Zhao D, Zhou B. In-domain GAN inversion for real image editing. In: Proceedings of the 16th European Conference on Computer Vision. 2020, 592–608
58. Patashnik O, Wu Z, Shechtman E, Cohen-Or D, Lischinski D. StyleCLIP: text-driven manipulation of StyleGAN imagery. In: Proceedings of 2021 IEEE/CVF International Conference on Computer Vision. 2021, 2085–2094
59. Radford A, Kim J W, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I. Learning transferable visual models from natural language supervision. In: Proceedings of the 38th International Conference on Machine Learning. 2021, 8748–8763
60. Gu S, Chen D, Bao J, Wen F, Zhang B, Chen D, Yuan L, Guo B. Vector quantized diffusion model for text-to-image synthesis. In: Proceedings of 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, 10696–10706
61. Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-resolution image synthesis with latent diffusion models. In: Proceedings of 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022, 10684–10695
62. Ramesh A, Dhariwal P, Nichol A, Chu C, Chen M. Hierarchical text-conditional image generation with CLIP latents. 2022, arXiv preprint arXiv: 2204.06125
63. Saharia C, Chan W, Saxena S, Lit L, Whang J, Denton E L, Ghasemipour S K S, Ayan B K, Mahdavi S S, Gontijo-Lopes R, Salimans T, Ho J, Fleet D J, Norouzi M. Photorealistic text-to-image diffusion models with deep language understanding. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. 2022, 2643
64. Balaji Y, Nah S, Huang X, Vahdat A, Song J, Zhang Q, Kreis K, Aittala M, Aila T, Laine S, Catanzaro B, Karras T, Liu M Y. eDiff-I: text-to-image diffusion models with an ensemble of expert denoisers. 2022, arXiv preprint arXiv: 2211.01324
65. Wallace B, Gokul A, Naik N. EDICT: exact diffusion inversion via coupled transformations. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 22532–22541
66. Ruiz N, Li Y, Jampani V, Pritch Y, Rubinstein M, Aberman K. DreamBooth: fine tuning text-to-image diffusion models for subject-driven generation. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 22500–22510
67. Gupta T, Kembhavi A. Visual programming: compositional visual reasoning without training. In: Proceedings of 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2023, 14953–14962
68. Liu H, Li C, Li Y, Lee Y J. Improved baselines with visual instruction tuning. In: Proceedings of 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2024
69. Schuhmann C. Improved aesthetic predictor, 2022. GitHub repository
70. Liu S, Zeng Z, Ren T, Li F, Zhang H, Yang J, Jiang Q, Li C, Yang J, Su H, Zhu J, Zhang L. Grounding DINO: marrying DINO with grounded pre-training for open-set object detection. In: Proceedings of the 18th European Conference on Computer Vision. 2025

Tiankai HANG received the BE degree from Southeast University, Nanjing, China in 2020. He is currently pursuing the PhD degree with the School of Computer Science and Engineering, Southeast University, China. He is also a long-term researcher intern at Microsoft Research Asia (MSRA). His research interests include computer vision, visual generation, multi-modal representation learning, and machine learning.



Shuyang GU is currently a Researcher in Visual Computing Group at Microsoft Research Asia (MSRA). He received his BS and PhD degrees from University of Science and Technology of China (USTC), China in 2017 and 2022, supervised by Prof. Yong Wang and Prof. Baining Guo. His research interests mainly focus on generative models, especially the theory and practical applications of Generative Adversarial Networks and diffusion models.

Dong CHEN received the BS and PhD degrees from the University of Science and Technology of China, China in 2010 and 2015, respectively. In 2015, he joined Microsoft Research. He is currently the Principal Researcher Manager of the Visual Computing Group with Microsoft Research Asia, China. He has authored or coauthored more than 50 papers in international conferences such as CVPR/ICCV/ECCV and holds 8 patents. His team is engaged in research on image synthesis models such as generative adversarial networks, denoising diffusion probabilistic model, and generative artificial intelligence. Multiple research results have been used in products such as Microsoft Cognitive Services, Windows Hello face unlock in Windows 10, and Microsoft Designer.

Xin GENG is a Chair Professor of Southeast University, China, Executive Vice Dean of the Graduate School, and Director of Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications, Ministry of Education, China. He previously served as Dean of the School of Computer Science and Engineering, the School of Software, and the Executive Dean of the School of Artificial Intelligence. He is a recipient of the National Science Fund for Distinguished Young Scholars and the Excellent Young Scientists Fund, and a Distinguished Fellow of the International Engineering and Technology Institute (IETI). His research primarily focuses on machine learning, pattern recognition, and computer vision, and he has published over 150 papers in leading international academic journals and conferences in these fields. He has received several prestigious awards, including the Second Prize of the National Natural Science Award, the First Prize of the National Teaching Achievement Award, the First Prize of the Ministry of Education Natural Science Award, and the Science Exploration Award.

Baining GUO (Fellow, IEEE) received the BS degree from Peking University, China, and the MS and PhD degrees from Cornell University, USA. He is currently a distinguished scientist of Microsoft Corporation. He is deputy managing director of Microsoft Research Asia, where he works on computer graphics, computer vision, and video analysis. Prior to joining Microsoft Research in 1999, he was a senior staff researcher with Intel Research in the Silicon Valley. He is a fellow of the ACM and Canadian Academy of Engineering.