Knowledge-Centric Agents for Workflow Generation in ComfyUI

Zhendong Li [AI], Lei Sun [AI,✉], Ruibo Ming [AI], He Zhang [♢], Danda Pani Paudel [AI], Luc Van Gool [AI], and Jinjin Gu [AI,✉]

[AI] INSAIT, Sofia University "St. Kliment Ohridski", [♢] Adobe Research
{zhendong.li, lei.sun, ruibo.ming, danda.paudel, luc.vangool, jinjin.gu}@insait.ai, hezhan@adobe.com
✉ *Corresponding authors.*

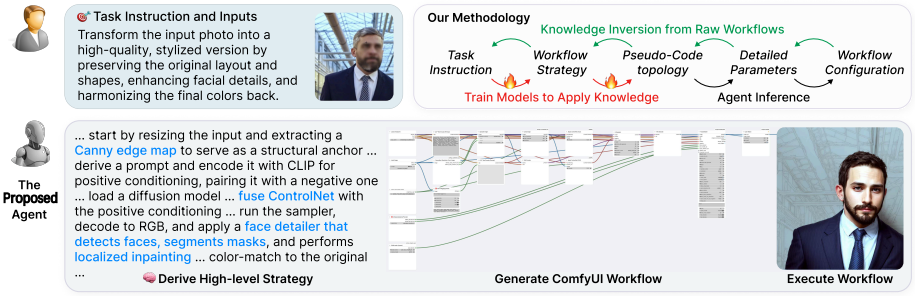

Fig. 1: Given a user task instruction, our agent learns to construct and execute a complete ComfyUI workflow. Through knowledge inversion, it distills pseudo-code structures and workflow strategies from large collections of real workflows. During inference, During inference, the agent executes the generated workflow to produce a high-quality visual result. The example shows how the agent interprets a stylistic editing request, derives a high-level strategy, composes the corresponding ComfyUI graph, and executes it to generate the final output.

Abstract. Workflow generation in visual creation systems such as ComfyUI demands not only syntactic accuracy but also expert-level reasoning over modular compositions. Existing large language model (LLM) approaches often treat this as a direct text-to-JSON generation task, struggling with structural brittleness and lacking the experiential knowledge required for effective design. We argue that successful workflow generation requires modeling knowledge itself, including its structure, hierarchy, and reasoning dynamics. To this end, we propose a knowledge-centric framework that learns to **invert**, **inject**, and **infer with knowledge** across multiple abstraction levels. We first perform knowledge inversion to distill hierarchical representations, ranging from full pseudo-codes and skeletons to high-level strategies, from large collections of real-world workflows. We then conduct knowledge injection through supervised fine-tuning, teaching the model to reason from task descriptions to strategies and from strategies to executable structures. During inference, the model performs reversible reasoning to synthesize executable

workflows, augmented by self-refinement for structural coherence. Extensive experiments demonstrate that our method produces workflows with richer node diversity, more coherent structures, and higher execution success rates than existing systems, establishing a new foundation for knowledge-driven, agentic workflow generation.

1 Introduction

The whirlwind of progress in modern computer vision models has steadily transformed controllable image generation, creation, editing, and processing from laboratory prototypes into powerful tools capable of addressing complex real-world demands. In practice, both technical experts and artists often need to combine and coordinate multiple model components to accomplish those tasks. This results in a non-monolithic, modular composition paradigm for problem solving, where different modules are orchestrated into a Directed Acyclic Graph (DAG), commonly referred to as a workflow, and executed sequentially or conditionally. ComfyUI [8] embodies this philosophy and has become a widely adopted visual programming framework for building such workflows in image creation pipelines. Developing intelligent agents that can automatically generate and refine these workflows is thus not only of great practical relevance but also poses deep scientific challenges.

A ComfyUI workflow can be fully represented as a JSON configuration, which specifies all nodes, parameters, and interconnections. Several recent studies have attempted to leverage large language models (LLMs) to generate these configurations automatically [13, 14, 16, 30, 35, 37]. However, generating valid and effective workflows differs fundamentally from conventional code or text generation. It requires multi-level reasoning that spans from abstract conceptual design to concrete syntactic realization, each demanding distinct forms of knowledge and representation.

At the strategic level, workflow design is an open-ended and experience-driven process rather than a deterministic programming task. ComfyUI provides a rich ecosystem of node types and model variants, each exhibiting complex, often nonlinear behaviors. Their combinations yield an immense compositional space, where small structural changes can produce radically different outcomes. Human designers rely heavily on tacit expertise, iterative experimentation, and intuition to determine whether segmentation is needed, how to arrange processing branches, or which control modules should interact. Such heuristic reasoning, grounded in experiential knowledge, remains highly challenging for LLMs to acquire or reproduce purely from textual data.

At the structural level, even when the high-level strategy is known, translating it into a syntactically correct and executable JSON configuration poses significant challenges. Workflows are large discrete graphs with intricate input–output dependencies. The agent must not only infer the correct topology but also instantiate valid node parameters and maintain type-safe data flow. The brittleness of JSON syntax, coupled with the combinatorial explosion of possible graph struc-

tures, makes direct text-to-configuration generation highly unreliable. Small mistakes such as misconnected links or inconsistent parameter bindings can easily make the workflow fail to execute.

These challenges collectively reveal a deeper limitation: *existing language models lack structured, multi-level knowledge about workflow composition.* While they excel at surface-level pattern completion, they struggle to reason about the hierarchical dependencies, causal logic, and experiential heuristics that underlie expert workflow design. To overcome this gap, we propose a knowledge-centric framework that explicitly reconstructs, injects, and utilizes such multi-level representations.

Our approach begins by **inverting knowledge** from large collections of real workflows to extract hierarchical representations of strategy, structure, and reasoning; then **injects** this distilled knowledge into language models through supervised fine-tuning; and finally **infers with knowledge**, performing reversible reasoning from task descriptions to executable workflows. This unified paradigm transforms workflow generation from an ad-hoc text generation problem into a principled process of knowledge reasoning, paving the way for agentic systems capable of composing complex visual pipelines with expert-level reliability. In practice, we curate a collection of high-quality ComfyUI workflows to serve as both training and evaluation data. Experimental results demonstrate that our method produces workflows with richer node diversity, more complex yet coherent structures, and more accurate connectivity. Compared with existing approaches, it achieves a higher proportion of successfully solved tasks.

2 Related Work

ComfyUI and Automatic ComfyUI Assistant. As AI-generated content (AIGC) continues to advance [11, 15], an increasing number of models and tasks, including text-to-image generation, controllable synthesis, and image editing, are emerging [18, 20, 24, 41]. This shift has moved the community from relying on a single model to composing multiple models to accomplish more complex and customized workflows. Consequently, ComfyUI [8] has become increasingly popular for its flexible, node-based design. ComfyUI connects modular function nodes into directed acyclic workflows, enabling users to build and reuse complex AIGC pipelines. However, these workflows can quickly become complex, often involving nodes for prompt refinement, foreground–background segmentation, super-resolution, and many other operations, which creates a strong demand for automated workflow design tools. To address this need, a variety of automated workflow-design tools have emerged [6,13,14,16,30,35–37]. ComfyUI-Copilot [36] assists users in selecting appropriate parameters and models; ComfyGen [13] focuses specifically on text-to-image generation; ComfyBench [37] and ComfyMind [14] adopt template-based composition strategies that cover most common tasks, with the latter offering tree-structured rollback. Meanwhile, ComfyGPT [16] and ComfyUI-R1 [35] leverage supervised fine-tuning and GRPO [25] reinforcement learning to inject workflow-design knowledge into LLMs. How-

ever, when new workflows emerge from the community, these methods must be retrained, limiting their flexibility and scalability for future extensions.

LLMs and LLM-Agent. LLMs have rapidly advanced natural language understanding and generation. BERT [10] introduced bidirectional masked language modeling, greatly improving contextual representation learning. The GPT series [3] shifted toward large-scale autoregressive pretraining, enabling few-shot and in-context learning. Later, ChatGPT and GPT-4 [1] combined instruction tuning and RLHF to enhance alignment and interaction quality. Recent models such as LLaMA [33], Qwen [38], Gemini [7, 32] further expand reasoning, efficiency, and multimodal capabilities. Building on these foundations, LLM-powered agents have emerged as autonomous entities capable of perceiving their environments, planning over extended horizons, and interacting with diverse external tools [1, 12]. Compared with traditional symbolic or RL-based agents [17, 28], those driven by LLMs exhibit superior adaptability, compositional reasoning, and task generalization [4, 19]. Recent research has further enhanced their autonomy through graph-based reasoning [2, 39], dynamic tool utilization [21,23,26,34], and self-reflective learning mechanisms [27]. Some studies further explore automated optimization of agentic workflows. For example, GPTSwarm [42] models language agents as nodes in an optimizable graph, while TextGrad [40] introduces gradient-like optimization to iteratively refine prompts and agent behaviors. In addition, recent work investigates scalable evaluation strategies for agent systems [43], while [5] highlights potential biases in automated evaluation. These LLM-agents are applied widely in the areas of code generation [9], health care [22, 29], and scientific discovery [31]. Moreover, as a challenging and complex problem, workflow generation is also well-suited to be addressed by LLM agents, which can decompose tasks, plan multi-step actions, and coordinate across tools.

3 Method

3.1 Overview

Generating a ComfyUI workflow is a complex problem with several layers of reasoning and representation. A workflow is encoded as a JSON graph where nodes are functional modules and edges describe data or control dependencies. Hence, workflow generation is equivalent to producing a valid and executable JSON file, which is far more delicate than ordinary text or code generation.

This complexity arises across several levels. Human experts do not jump from a task instruction directly to JSON. They first form a high level strategy that decides the overall processing plan. Typical decisions include whether segmentation is required, whether branches should process content separately, how many controls to introduce, and how controls interact. This high-level strategy requires substantial domain knowledge and intuitive understanding of visual models and their compositional logic.

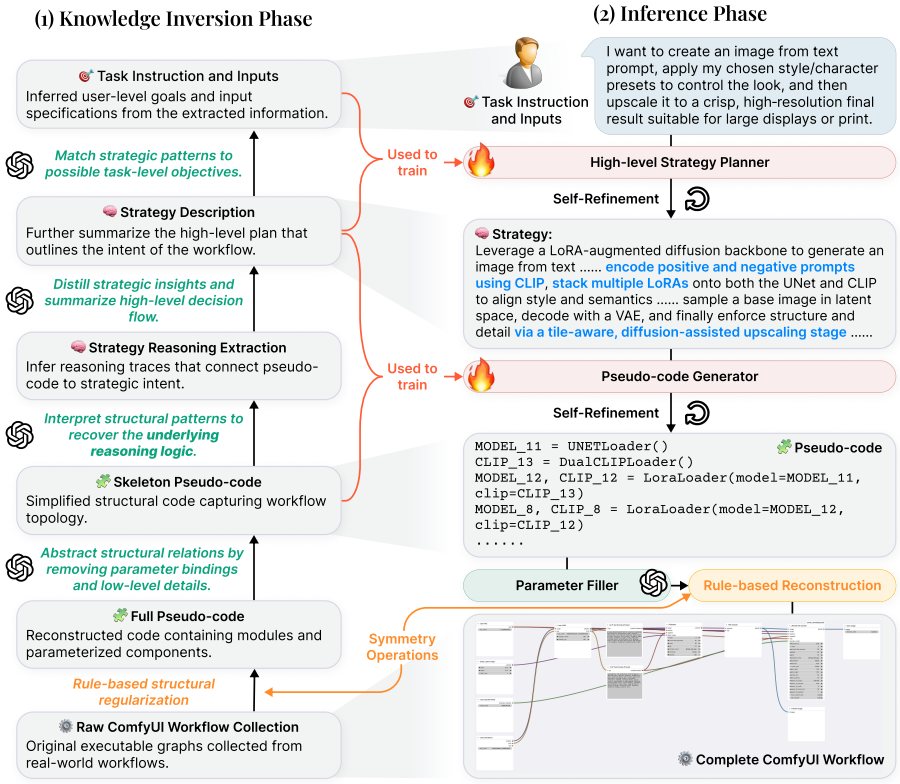

Fig. 2: Overview of our framework. The system consists of two symmetric phases. (1) Knowledge Inversion distills hierarchical representations from real ComfyUI workflows. (2) Knowledge Inference reverses this process to generate workflows from instructions via a strategy planner, pseudo-code generator, and rule-based reconstruction.

Given the strategy, experts construct the workflow topology, that is, a structured connection of modules. This step is also difficult because the graph can be very large, discrete, and complex. Directly generating JSON connections from text is extremely hard for language models due to syntactic brittleness and combinatorial explosion. After the topology is formed, parameter tuning and detailed module configurations can be carried out, which are comparatively more rule-driven or pattern-based. Therefore, workflow generation is a multi-level process, where each level poses distinct reasoning and learning challenges. For example, translating topology into a full workflow can be rule-based, but deriving the topology itself from task instructions and strategies demands substantial knowledge and reasoning ability.

The key idea of our approach is to inverse knowledge, inject knowledge, and infer with knowledge across these hierarchical levels. We systematically learn

from large collections of real-world workflows, distill the underlying knowledge representations, and integrate them into a reasoning-capable agent system.

3.2 Knowledge Inversion

The goal of this stage is to inverse raw and heterogeneous workflow data into a set of hierarchical knowledge representations that can be effectively learned by language models or agent systems. As illustrated in Figure 2 (1), the entire process proceeds from the bottom to the top, progressively abstracting and reorganizing knowledge contained in large collections of raw workflows.

- **From raw workflow to full pseudo-code.** We first employ a *rule-based regularization* procedure to convert each workflow into a compact, machine-readable format called *Full Pseudo-code*. This representation preserves all module names, topological connections, and parameter values, while removing the syntactic and ComfyUI-specific styling information. This step does not involve any learning and serves as the structural foundation for higher-level abstraction.
- **To skeleton pseudo-code.** The next step abstracts the low-level implementation details from the Full Pseudo-code. A language model removes parameter bindings and numerical values while maintaining the functional topology of the workflow. The result is a *Skeleton Pseudo-code* that captures only the essential graph structure and execution flow, providing a clean target for structure-aware learning.
- **To strategy reasoning.** Given the abstracted skeleton, we use large language models to perform semantic interpretation to infer the underlying reasoning traces that explain *why* the workflow is organized in such a way. This *Strategy Reasoning Extraction* step reconstructs the author's decision-making process. For instance, why multiple controls are fused, why branches are separated, or why a face-detailing module is placed after denoising. The output is a structured reasoning trace that links topological motifs to functional purposes.
- **To high-level strategy.** The inferred reasoning trace is then summarized into a concise *Strategy Description*, which represents the high-level processing plan that an expert might verbally describe. This level captures the overall solution logic of the workflow, such as whether segmentation is required, which modules should cooperate, and how controls interact.
- **To task instruction and inputs.** Since most collected workflows lack explicit user prompts, we further infer the corresponding *task instruction and input specifications* required for training. This is achieved by aligning the generated high-level strategy with retrieved exemplars and by decoding plausible task descriptions consistent with the structural and semantic context of the workflow. The result is a natural-language-level supervision signal that reconstructs the user intent behind each workflow.

This hierarchical inversion provides structured knowledge that can later be injected into large language models and utilized by agentic reasoning frameworks

for workflow synthesis. Each level can serve as either a hard target or a weak supervision signal for downstream training.

3.3 Knowledge Injection

After the hierarchical knowledge representations are distilled through inversion, the next stage is to inject them into language models so that the models can utilize this knowledge during inference. Not all levels require knowledge injection. Some transformations, such as filling in detailed parameters or expanding a skeleton pseudo-code into a complete version, can be handled by existing model priors or simple rule-based completion. However, transitions that involve reasoning, creativity and experiments, such as mapping from a task description to a strategy, or from a strategy to a new pseudo-code, demand explicit knowledge injection. We therefore categorize the knowledge injection process according to its learning necessity:

- **Task → Strategy.** This transition encodes the experiential knowledge of human experts, illustrating how they analyze a task, decompose it into subgoals, and select processing modules accordingly. Since this reasoning involves tacit expertise rather than explicit syntax, it cannot be inferred reliably by general-purpose models. Supervised fine-tuning (SFT) is therefore essential to inject this experience-driven knowledge.
- **Strategy → Skeleton Pseudo-code.** Translating a high-level plan into executable structures requires compositional reasoning and graph construction skills that the base model lacks. We again employ SFT to teach the model to generate structured, syntactically valid pseudo-codes consistent with the strategy's intent. This step enables the model to internalize topological patterns and causal relations between modules.
- **Skeleton → Full Pseudo-code.** This level mainly concerns parameter restoration and stylistic completion, which can be achieved through rule-based expansion or lightweight prompting. It does not require additional supervised training.

We emphasize that SFT is the primary mechanism for knowledge injection, as it directly teaches the model to produce new reasoning patterns or representations beyond its pretraining scope. By contrast, reinforcement-style alignment methods such as GRPO are mainly effective for local preference optimization within an already feasible generation space. In our setting, where the goal is to endow the model with new capabilities such as planning, decomposition, and workflow construction, SFT serves as the primary knowledge carrier. Some studies indeed demonstrate that in similar settings, SFT continues to play the dominant role for capability injection [35].

In practice, we fine-tune a large language model, Qwen3-14B [38], on paired examples of *(task, strategy)* and *(strategy, pseudo-code)*. During training, we also introduce an auxiliary objective of *strategy reasoning extraction*, which requires the model to generate intermediate reasoning traces explaining each strategic

decision. This auxiliary task regularizes the model to learn the discourse and logical structure of expert reasoning, thereby improving both interpretability and generalization during workflow synthesis.

3.4 Knowledge Inference

Once the language model has been equipped with the injected knowledge, inference on new tasks becomes essentially the reverse process of knowledge inversion, as illustrated in Figure 2 (2). Given a user-provided task instruction and inputs, the system first invokes a *High-Level Structure Planner*, an SFT-trained model that performs strategic reasoning to produce a high-level strategy describing the overall processing logic, decomposition, and control structure of the workflow. The strategy reflects the distilled experiential knowledge obtained from the large corpus of real workflows. Next, the generated strategy is passed to a *Pseudo-code Generator*, another SFT-trained model that converts the abstract plan into an executable skeleton pseudo-code. A pretrained model is then employed to fill in parameters and bindings based on the contextual semantics of each module, restoring a full pseudo-code with appropriate configuration values. Finally, the rule-based reconstruction procedure converts the full pseudo-code into a fully functional ComfyUI workflow, which can be directly executed or visualized in the target system.

To further improve robustness, we incorporate self-refinement mechanisms after both the strategy planning and pseudo-code generation stages. The model is prompted to re-evaluate and refine its own outputs, correcting incomplete reasoning steps and enhancing structural coherence. This iterative refinement leads to more complete, logically consistent, and executable workflows.

3.5 Discussion with Prior Works

Across the growing body of research on ComfyUI [35] workflow generation, there exists a common aspiration: to reduce the inherent complexity of workflow construction and to inject domain knowledge that enables structured reasoning. Yet the ways the prior works operationalize this goal differ.

Early frameworks such as ComfyAgent [37] and ComfyMind [14] approach the challenge by introducing constrained abstractions. They replaced raw node-level graphs with meta-nodes whose internal connectivity was predefined, lowering combinatorial complexity. This approach, while controllable, inevitably limited expressiveness by constraining the solution space to handcrafted schemas.

Subsequent learning-based systems, including ComfyGPT [16] and ComfyUI-R1 [35], shifted the focus from symbolic simplification to data-driven capability. Instead of fixing structural templates, they trained language or reasoning models on large collections of workflows to predict connections or generate pseudo-codes directly from textual descriptions. However, their knowledge remained implicitly entangled within model parameters, making it difficult to interpret, adapt, or reuse across levels of abstraction.

Our framework departs from both directions. Rather than simplifying the problem or burying knowledge inside opaque model weights, we make knowledge itself the central representational currency. Through knowledge inversion, we decompose complex workflows into hierarchical representations that capture structure, strategy, and reasoning. This paradigm does not merely reduce complexity; it reorganizes it by transforming the implicit experiential knowledge of experts into explicit, learnable, and reusable forms. In doing so, it moves workflow generation from ad-hoc pattern learning toward a principled, knowledge-centric foundation for agentic reasoning.

4 Experiments

4.1 Settings

Workflow Collection. We collected over 10,000 workflow instances from the Internet, covering a wide range of task types and node structures. However, the raw data were highly noisy and redundant. To construct a high-quality, diverse, and semantically consistent dataset, we performed large-scale structural deduplication and filtering. Specifically, we first identified potentially redundant workflows by comparing node-level structural differences. When the ratio of differing nodes between two workflows was below 10%, they were considered structurally similar. In such cases, we employed GPT-4o to evaluate semantic and procedural equivalence, retaining only the version with more standardized and widely used node types. We then merged fully identical workflows by unifying their input entries and removed workflows that were either too simple (fewer than six nodes) or excessively complex (more than ninety nodes). Task descriptions were further normalized and consolidated to eliminate near-duplicate objectives while maintaining structural and semantic diversity.

After multi-stage cleaning and filtering, we obtained a curated dataset of 912 high-quality workflows spanning diverse task distributions and node types. Among them, 882 workflows were used for training, while 30 workflows and their corresponding task queries were reserved for testing. The test workflows were grouped into three categories based on their similarity to the training data: *Challenge 1* (high similarity), *Challenge 2* (medium similarity), and *Real-world Cases* (no comparable training samples). Similarity was assessed by a GPT-based evaluator across four dimensions: task objective, technical method, structural form, and application context.

In addition to this split, we further evaluate our method on two external benchmarks to assess generalization ability. First, we evaluate on the FlowBench dataset [16] using a test set of 30 tasks. Second, we conduct evaluation on a subset of ComfyBench consisting of 20 tasks sampled from the *Creative* and *Complex* categories. Together, these evaluations cover both in-domain and out-of-domain workflow generation scenarios.

Compared Methods. Following [37], we adopt six representative workflow-generation paradigms for comparison: Zero-Shot, Few-Shot, Chain-of-Thought

(CoT), CoT with Self-Consistency (CoT-SC), Retrieval-Augmented Generation (RAG), and ComfyAgent. These methods correspond to the variants evaluated in Table 1, 2, 3, 4. To ensure fairness, all the methods use GPT-5 as the underlying LLM interface, consistent with the configuration of our refine agent. More details of these baseline methods can be found in the supplementary material.

Implementation detail. We used the Qwen3-14B model as the backbone for SFT training. The model was trained on the two tasks described above. Both stages were fine-tuned using the LoRA technique on a single NVIDIA H200 GPU. During the instruction-to-strategy stage, we set the maximum sequence length to 2048 tokens, while in the strategy-to-pseudocode stage, it was extended to 4096 tokens. The LoRA configuration used a rank of 16 and a scaling factor (α) of 32. For the pre-trained large language model used for agent building, we adopt GPT-5 (gpt-5-turbo, 2025 release).

4.2 Evaluation Metrics

We comprehensively evaluate generated workflows from both *structural* and *semantic* perspectives using four complementary metrics: Valid Node Diversity (VND), Node Completeness (NodeComp), Link Completeness (LinkComp), and Task Consistency (TaskCons).

VND quantifies the diversity of valid node types present in generated workflows, reflecting the model's ability to produce varied and functionally valid module compositions rather than relying on repetitive or trivial components.

NodeComp measures the proportion of valid nodes that are correctly instantiated and involved in at least one connection. A node is considered valid if its type is registered, non-empty, and linked to other nodes. The ratio of valid to total nodes indicates the semantic and functional completeness of the workflow.

LinkComp evaluates the structural correctness of node connections. For each node, input–output consistency is verified: all declared inputs must exist and correctly reference valid outputs, and all output slot indices must remain within bounds. The proportion of nodes passing these checks represents the workflow's topological soundness.

TaskCons assesses how well the generated workflow fulfills the target task described in the instruction. A GPT-based evaluator scores how well each workflow fulfills its task (e.g., "upscale", "relighting", "pose-consistent animation"). Higher scores indicate better alignment between the task intent and the workflow's functional structure.

In addition, we evaluate executability and task fulfillment through two outcome-level indicators: **Pass** and **Resolve**. A workflow is counted as *Pass* if it executes successfully in ComfyUI without structural or runtime errors, and as *Resolve* if it additionally produces an output that fulfills the intended task goal. Together, these indicators assess both syntactic validity and the functional effectiveness of workflow generation. To ensure stable and unbiased evaluation, we do not rely on LLM-based judges. Instead, all workflows are executed manually in ComfyUI and the outputs are assessed through human inspection.

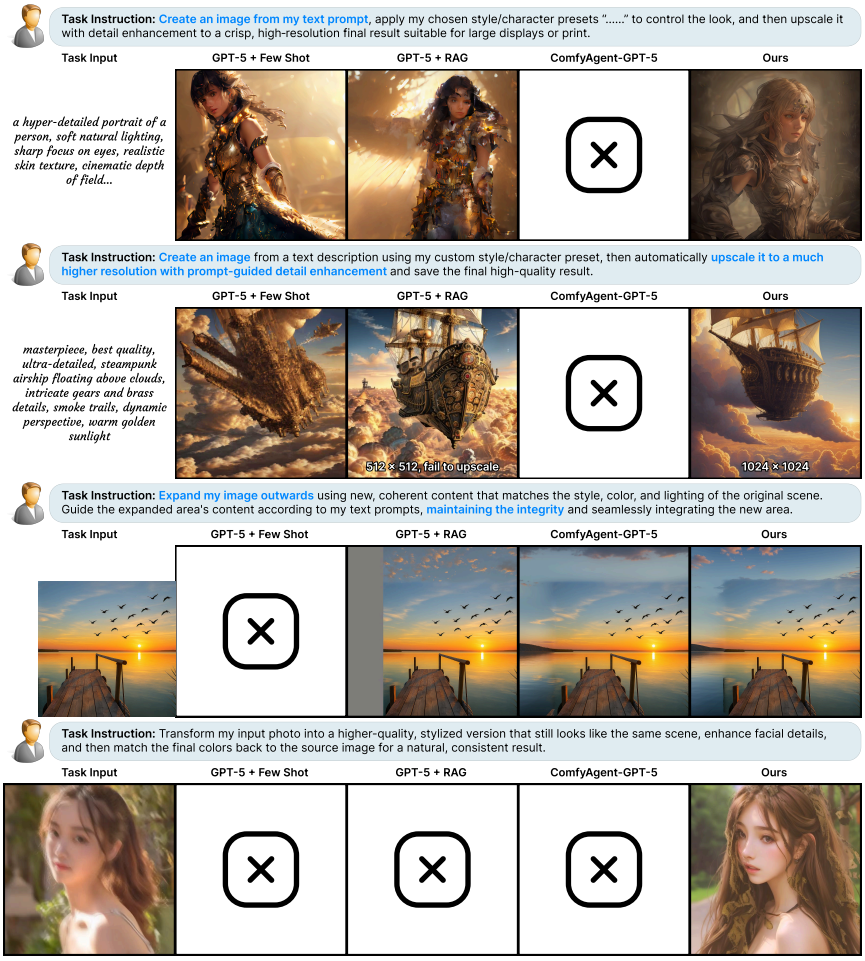

Fig. 3: Comparison with baseline methods. Given a user instruction, our method accurately adheres to the task description.

4.3 Results

Quantitative evaluation. We evaluate workflow-generation quality on our test set using four complementary metrics: VND, NodeComp, LinkComp, and TaskCons. Together, these metrics measure workflow diversity, structural completeness, and task-level correctness. As shown in Table 1, our method achieves clear improvements across all metrics. Our VND score reaches 152, more than 3.6 times the highest baseline score of 42, indicating that our system can compose substantially richer and more diverse workflow structures than template-based methods.

Our method also attains markedly higher structural and semantic accuracy than all the other methods. Its NodeComp, LinkComp, and TaskCons scores are

Table 1: Comparison of workflow generation performance on the test set. VND: Valid Node Diversity. NodeComp: Node Completeness. LinkComp: Link Completeness. TaskCons: Task Consistency.

Method	VND↑	NodeComp (%)↑	LinkComp (%)↑	TaskCons (%)↑
GPT-5 + Zero-Shot	0.0	0.0	0.0	0.0
GPT-5 + Few-Shot	20	36.7	36.7	34.7
GPT-5 + CoT	26	33.3	33.3	32.9
GPT-5 + CoT-SC	21	36.7	36.7	34.8
GPT-5 + RAG	42	56.7	56.7	53.6
GPT-5 + ComfyAgent	40	56.7	56.7	55.1
Ours	**152**	**96.3**	**98.3**	**86.9**

Table 2: Results grouped by test set categories. *Pass* indicates the generated workflow executes successfully without structural or runtime errors. *Resolve* indicates the workflow executes and fulfills the given instruction.

Method	**Challenge 1**		**Challenge 2**		**Real-World Cases**		**Average**	
	Pass (%)	Resolve (%)	Pass (%)	Resolve (%)	Pass (%)	Resolve (%)	Pass (%)	Resolve (%)
GPT-5 + Zero-Shot	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
GPT-5 + Few-Shot	41.6	25.0	37.5	25.0	0.0	0.0	26.4	16.7
GPT-5 + CoT	33.3	25.0	25.0	12.5	10.0	10.0	22.8	15.8
GPT-5 + CoT-SC	25.0	25.0	37.5	12.5	10.0	0.0	24.2	12.5
GPT-5 + RAG	66.7	41.7	37.5	25.0	40.0	10.0	48.1	25.6
GPT-5 + ComfyAgent	41.7	33.3	37.5	25.0	30.0	30.0	36.4	29.4
Ours	**83.3**	**58.3**	**87.5**	**75.0**	**90.0**	**50.0**	**86.9**	**61.1**

consistently far above GPT-5-based methods and ComfyAgent, demonstrating that the generated workflows are both well-formed and aligned with the intended task logic. These results show that template-based or retrieval-driven methods are fundamentally limited, while our model can generate diverse, structurally sound, and task-faithful workflows at scale.

As reported in Table 2, we further evaluate workflow executability within ComfyUI on our test set. Our approach achieves the highest pass and solve rates across all test categories—Challenge 1, Challenge 2, and Real-World Cases. For example, the average Pass rate reaches 86.9%, compared with 36.4% for ComfyAgent, and Resolve improves from 29.4% to 61.1%. These results indicate that our workflows are not only theoretically valid but also executable in practice and able to fulfill the specified task instruction. The strong performance on real-world cases further highlights the robustness of the generated pipelines.

We evaluate workflow-generation quality on ComfyBench [37], a benchmark of real-world visual-creation workflows that enables assessing model generalization. The evaluation is conducted on a held-out set of 20 tasks randomly sampled from the *Creative* and *Complex* categories. As shown in Table 3, our method substantially outperforms all the other methods across both structural and execution metrics. In particular, our model achieves a VND score of 138, which

Table 3: Overall workflow generation performance on ComfyBench. Workflow structure metrics include VND (Valid Node Diversity), NodeComp (Node Completeness), LinkComp (Link Completeness), and TaskCons (Task Consistency). Execution metrics report the average Pass and Resolve rates.

	Workflow Structure				Execution	
Method	VND↑	NodeComp (%)↑	LinkComp (%)↑	TaskCons (%)↑	Pass (%)↑	Resolve (%)↑
GPT-5 + Zero-Shot	0	0.0	0.0	0.0	0.0	0.0
GPT-5 + Few-Shot	14	35.0	35.0	29.8	10.0	5.0
GPT-5 + CoT	12	25.0	25.0	19.5	5.0	0.0
GPT-5 + CoT-SC	18	20.0	20.0	17.1	0.0	0.0
GPT-5 + RAG	45	60.0	60.0	57.7	40.0	25.0
GPT-5 + ComfyAgent	43	65.0	65.0	59.3	35.0	20.0
Ours	**138**	**79.8**	**82.7**	**85.0**	**50.0**	**25.0**

is over 3× higher than the strongest baseline (43–45), indicating significantly richer workflow compositions. Structural correctness also improves markedly, with NodeComp increasing from 65.0% to 79.8% and LinkComp from 65.0% to 82.7%, while TaskCons rises from 59.3% to 85.0%. These structural gains translate into stronger execution reliability, yielding 50.0% Pass and 25.0% Resolve on average across the benchmark. Overall, these results show that our knowledge-centric framework generates workflows that are substantially more diverse, structurally coherent, and executable, demonstrating strong generalization on real-world workflow generation tasks.

We evaluate our method on the FlowBench [16] using a test set of 30 tasks, as summarized in Table 4. Our approach consistently outperforms all all the other methods across both workflow structure and execution metrics. In particular, it achieves a VND score of 145, substantially higher than the strongest baseline (41–42), indicating significantly richer workflow compositions. Structural correctness is also markedly improved, with NodeComp reaching 94.4% and LinkComp 98.0%, compared to around 53–73% for existing methods, while TaskCons improves to 93.4%. These structural advantages translate into stronger execution reliability, achieving 66.7% Pass and 36.7% Solve rates, exceeding all baseline approaches. Overall, the results demonstrate that our method achieves strong generalization on the FlowBench [16] while consistently outperforming existing all the other methods.

LLM Judge Bias Analysis. To examine the robustness of task consistency evaluation, we compare two LLM judges, GPT-5 and Gemini-2.5, on the same set of generated workflows. As shown in Table 5, both evaluators produce highly consistent performance trends across prompting strategies. Our method achieves the highest task consistency under both GPT-5 (86.9%) and Gemini-2.5 (85.6%), while the relative ranking of competing approaches remains largely unchanged. This consistency suggests that the evaluation results are not sensitive to the choice of LLM judge and are unlikely to be dominated by model-specific bias. In addition, execution-based metrics such as Pass and Solve provide an independent signal of task correctness, further supporting the reliability of the evaluation.

Table 4: Overall performance comparison on FlowBench [16]. Read as Tab.3.

Method	Workflow Structure				Execution	
	VND↑	NodeComp (%)↑	LinkComp (%)↑	TaskCons (%)↑	Pass (%)↑	Solve (%)↑
GPT-5 + Zero-Shot	0	0.00	0.00	0.00	0.0	0.0
GPT-5 + Few-Shot	27	46.67	46.67	45.50	26.7	16.7
GPT-5 + CoT	24	43.33	43.33	42.10	23.3	16.7
GPT-5 + CoT-SC	26	40.00	40.00	38.20	20.0	13.3
GPT-5 + RAG	42	73.33	73.33	67.43	46.7	26.7
GPT-5 + ComfyAgent	41	53.33	53.33	50.80	40.0	26.7
Ours	**145**	**94.4**	**98.0**	**93.4**	**66.7**	**36.7**

Table 5: Task consistency (%) evaluated by two LLM judges (GPT-5 and Gemini-2.5) across different methods. Higher is better.

Model	Zero-Shot	Few-Shot	CoT	CoT-SC	RAG	ComfyAgent	Ours
Gemini-2.5	0	34.0	30.8	34.2	52.0	40.8	**85.6**
GPT-5	0	34.7	32.9	34.8	53.6	55.1	**86.9**

Table 6: Comparison with Qwen3-32B Q2P on workflow generation. Read as Tab. 3

Method	Workflow Structure			Execution	
	VND↑	NodeComp (%)↑	LinkComp (%)↑	Pass (%)↑	Solve (%)↑
Qwen3-32B Q2P	148	92.3	95.8	56.7	33.3
Ours	**152**	**96.3**	**98.3**	**86.9**	**61.1**

Effect of Hierarchical Workflow Modeling. To provide a strong training-based baseline, we train Qwen3-32B end-to-end to directly map user instructions to pseudocode workflows (Q2P). As shown in Table 6, our method with Qwen3-14B consistently outperforms the Qwen3-32B Q2P baseline across both workflow structure and execution metrics, despite using a smaller backbone model. In particular, our approach improves VND from 148 to 152, NodeComp from 92.3% to 96.3%, and LinkComp from 95.8% to 98.3%, indicating more complete and structurally coherent workflow generation. The advantage becomes even more pronounced in execution reliability: Pass increases from 56.7% to 86.9% and Solve from 33.3% to 61.1%, demonstrating that our method produces workflows that are substantially more executable and task-consistent. These results highlight the benefit of the proposed hierarchical design compared with direct end-to-end training. Instead of learning a single mapping from instructions to workflows, our framework decomposes workflows into multi-level supervision signals, including high-level design strategies, intermediate reasoning traces, and structural skeleton topologies. This structured supervision enables the model to capture workflow design principles explicitly and to compose modules in a more systematic manner, leading to better structural fidelity and higher execution success rates.

Qualitative evaluation. As shown in Fig. 3, our method accurately interprets concise task instructions and assembles suitable node compositions across diverse tasks, including text-to-image generation, outpainting, face restoration, and style transfer. The qualitative results show that our agent generates more complete and better-structured workflows, producing higher-quality visual outputs than ComfyAgent and the other methods.

Limitations and Discussion. We observe a limitation when tasks require relatively rare node types. In such cases, the model tends to substitute them with more common nodes, which may break functional dependencies and increase structural errors in the generated workflow. We attribute this behavior mainly to limited coverage of rare nodes in the training data; expanding the diversity and scale of training workflows could help mitigate this issue. In addition, incorporating reinforcement-style optimization (e.g., GRPO) may further improve robustness.

5 Conclusion

In this work, we present a knowledge-centric framework for workflow generation in visual creation systems such as ComfyUI. Unlike prior LLM-based approaches that treat workflow synthesis as a direct text-to-JSON prediction problem, our method explicitly models the structure, hierarchy, and dynamics of expert knowledge.Through knowledge inversion, the model learns multilevel representations, including pseudo-codes, structural skeletons, and design strategies, from large collections of real workflows. Knowledge injection then guides the model to map tasks to strategies and strategies to executable structures via supervised reasoning. During inference, reversible reasoning and self-refinement enable the generation of coherent, executable workflows. Extensive experiments show that our approach produces more diverse nodes, more consistent compositions, and significantly higher execution success rates than existing methods. Overall, this work establishes a new direction for knowledge-driven, agentic workflow generation in visual creation systems.

Acknowledgment

This research was partially funded by the Ministry of Education and Science of Bulgaria, through support for INSAIT as part of the Bulgarian National Roadmap for Research Infrastructure. This project is also partially supported by Adobe Research.

References

1. Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al.: Gpt-4 technical report. arXiv preprint arXiv:2303.08774 (2023)

2. Besta, M., Blach, N., Kubicek, A., Gerstenberger, R., Podstawski, M., Gianinazzi, L., Gajda, J., Lehmann, T., Niewiadomski, H., Nyczyk, P., et al.: Graph of thoughts: Solving elaborate problems with large language models. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 17682–17690 (2024)
3. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
4. Chen, G., Dong, S., Shu, Y., Zhang, G., Sesay, J., Karlsson, B.F., Fu, J., Shi, Y.: Autoagents: A framework for automatic agent generation. arXiv preprint arXiv:2309.17288 (2023)
5. Chen, G.H., Chen, S., Liu, Z., Jiang, F., Wang, B.: Humans or llms as the judge? a study on judgement bias. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. pp. 8301–8327 (2024)
6. Chen, J., Zhu, X., Wang, Y., Liu, T., Chen, X., Chen, Y., Leong, C.T., Ke, Y., Liu, J., Yuan, Y., et al.: Symbolic representation for any-to-any generative tasks. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 27816–27826 (2025)
7. Comanici, G., Bieber, E., Schaekermann, M., Pasupat, I., Sachdeva, N., Dhillon, I., Blistein, M., Ram, O., Zhang, D., Rosen, E., et al.: Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. arXiv preprint arXiv:2507.06261 (2025)
8. comfyanonymous: Comfyui. `https://github.com/comfyanonymous/ComfyUI` (2023)
9. Cursor: Cursor: Ai-powered code editor. `https://www.cursor.com/en`
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers). pp. 4171–4186 (2019)
11. Dhariwal, P., Nichol, A.: Diffusion models beat gans on image synthesis. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 8780–8794. Curran Associates, Inc. (2021), `https://proceedings.neurips.cc/paper_files/paper/2021/file/49ad23d1ec9fa4bd8d77d02681df5cfa-Paper.pdf`
12. Franklin, S., Graesser, A.: Is it an agent, or just a program?: A taxonomy for autonomous agents. In: International workshop on agent theories, architectures, and languages. pp. 21–35. Springer (1996)
13. Gal, R., Haviv, A., Alaluf, Y., Bermano, A.H., Cohen-Or, D., Chechik, G.: Comfygen: Prompt-adaptive workflows for text-to-image generation. arXiv preprint arXiv:2410.01731 (2024)
14. Guo, L., Xu, X., Wang, L., Lin, J., Zhou, J., Zhang, Z., Su, B., Chen, Y.C.: Comfymind: Toward general-purpose generation via tree-based planning and reactive feedback. arXiv preprint arXiv:2505.17908 (2025)
15. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 6840–6851. Curran Associates, Inc. (2020), `https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf`

16. Huang, O., Ma, Y., Zhao, Z., Wu, M., Ji, J., Zhang, R., Hu, Z., Sun, X., Ji, R.: Comfygpt: A self-optimizing multi-agent system for comprehensive comfyui workflow generation. arXiv preprint arXiv:2503.17671 (2025)
17. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M.: Learning agile and dynamic motor skills for legged robots. Science Robotics **4**(26), eaau5872 (2019)
18. Kumari, N., Zhang, B., Zhang, R., Shechtman, E., Zhu, J.Y.: Multi-concept customization of text-to-image diffusion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1931–1941 (2023)
19. Li, C., Gan, Z., Yang, Z., Yang, J., Li, L., Wang, L., Gao, J., et al.: Multimodal foundation models: From specialists to general-purpose assistants. Foundations and Trends® in Computer Graphics and Vision **16**(1-2), 1–214 (2024)
20. Li, T., Ku, M., Wei, C., Chen, W.: Dreamedit: Subject-driven image editing. arXiv preprint arXiv:2306.12624 (2023)
21. Li, X., Zou, H., Liu, P.: Torl: Scaling tool-integrated rl (2025), `https://arxiv.org/abs/2503.23383`
22. Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., Liu, T.Y.: Biogpt: generative pre-trained transformer for biomedical text generation and mining. Briefings in bioinformatics **23**(6), bbac409 (2022)
23. Qian, C., Acikgoz, E.C., He, Q., Wang, H., Chen, X., Hakkani-Tür, D., Tur, G., Ji, H.: Toolrl: Reward is all tool learning needs (2025), `https://arxiv.org/abs/2504.13958`
24. 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 the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 22500–22510 (2023)
25. Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y.K., Wu, Y., Guo, D.: Deepseekmath: Pushing the limits of mathematical reasoning in open language models (2024), `https://arxiv.org/abs/2402.03300`
26. Shen, Y., Song, K., Tan, X., Li, D., Lu, W., Zhuang, Y.: Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face. Advances in Neural Information Processing Systems **36** (2024)
27. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., Yao, S.: Reflexion: Language agents with verbal reinforcement learning. Advances in Neural Information Processing Systems **36** (2024)
28. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
29. Singhal, K., Azizi, S., Tu, T., Mahdavi, S.S., Wei, J., Chung, H.W., Scales, N., Tanwani, A., Cole-Lewis, H., Pfohl, S., et al.: Large language models encode clinical knowledge. Nature **620**(7972), 172–180 (2023)
30. Sobania, D., Briesch, M., Rothlauf, F.: Comfygi: Automatic improvement of image generation workflows. arXiv preprint arXiv:2411.14193 (2024)
31. Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., Stojnic, R.: Galactica: A large language model for science. arXiv preprint arXiv:2211.09085 (2022)
32. Team, G., Anil, R., Borgeaud, S., Alayrac, J.B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A.M., Hauth, A., Millican, K., et al.: Gemini: a family of highly capable multimodal models. arXiv preprint arXiv:2312.11805 (2023)

33. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
34. Wu, C., Yin, S., Qi, W., Wang, X., Tang, Z., Duan, N.: Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671 (2023)
35. Xu, Z., Wang, Y., Yang, X., Wang, L., Luo, W., Zhang, K., Hu, B., Zhang, M.: Comfyui-r1: Exploring reasoning models for workflow generation. arXiv preprint arXiv:2506.09790 (2025)
36. Xu, Z., Yang, X., Wang, Y., Hu, Q., Wu, Z., Wang, L., Luo, W., Zhang, K., Hu, B., Zhang, M.: Comfyui-copilot: An intelligent assistant for automated workflow development. arXiv preprint arXiv:2506.05010 (2025)
37. Xue, X., Lu, Z., Huang, D., Wang, Z., Ouyang, W., Bai, L.: Comfybench: Benchmarking llm-based agents in comfyui for autonomously designing collaborative ai systems. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 24614–24624 (2025)
38. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv preprint arXiv:2505.09388 (2025)
39. Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., Narasimhan, K.: Tree of thoughts: Deliberate problem solving with large language models. Advances in Neural Information Processing Systems **36** (2024)
40. Yuksekgonul, M., Bianchi, F., Boen, J., Liu, S., Huang, Z., Guestrin, C., Zou, J.: Textgrad: Automatic" differentiation" via text. arXiv preprint arXiv:2406.07496 (2024)
41. Zhang, L., Rao, A., Agrawala, M.: Adding conditional control to text-to-image diffusion models. In: CVPR (2023)
42. Zhuge, M., Wang, W., Kirsch, L., Faccio, F., Khizbullin, D., Schmidhuber, J.: Gptswarm: Language agents as optimizable graphs. In: Forty-first International Conference on Machine Learning (2024)
43. Zhuge, M., Zhao, C., Ashley, D., Wang, W., Khizbullin, D., Xiong, Y., Liu, Z., Chang, E., Krishnamoorthi, R., Tian, Y., et al.: Agent-as-a-judge: Evaluate agents with agents. arXiv preprint arXiv:2410.10934 (2024)