

Automatic Method Illustration Generation for AI Scientific Papers via Drawing Middleware Creation, Evolution, and Orchestration

Zhuoling Li^{1*}, Jiarui Zhang^{1*}, Ping Hu², Jason Kuen³, Jiuxiang Gu³,
Hossein Rahmani¹, and Jun Liu^{1**}

¹ Lancaster University, United Kingdom

{z.li81, j.zhang76, h.rahmani, j.liu81}@lancaster.ac.uk

² University of Electronic Science and Technology of China, China
chinahuping@gmail.com

³ Adobe Research, United States
{kuen, jigu}@adobe.com

Abstract. Method illustrations (MIs) play a crucial role in conveying the core ideas of scientific papers, yet their generation remains a labor-intensive process. Here, we take inspiration from human authors’ drawing practices and correspondingly propose **FigAgent**, a novel multi-agent framework for high-quality automatic MI generation. Our FigAgent distills drawing experiences from similar components across MIs and encapsulates them into reusable drawing middlewares that can be orchestrated for MI generation, while evolving these middlewares to adapt to dynamically evolving drawing requirements. Besides, a novel Explore-and-Select drawing strategy is introduced to mimic the human-like trial-and-error manner for gradually constructing MIs with complex structures. Extensive experiments show the efficacy of our method. Project page is here.

Keywords: Figure Generation · Multi-Modal Learning · Multi-Agent

1 Introduction

Method illustration (MI), a figure that conveys the workflow and core design insights of a proposed method, has become an indispensable component of modern AI scientific papers. Yet, creating high-quality MIs remains labor-intensive, often requiring hours or even days of iteration. Hence, automating this process is highly desirable and has attracted growing research attention [8, 23].

To enable automatic MI generation, a few recent attempts [33, 50, 57, 58] leverage image generation techniques [11, 28–30, 37] (e.g., Nano Banana Pro [11]) to generate MIs from textual prompts. Yet, despite the demonstrated performance in natural or artistic image synthesis, existing image generation techniques are

* Equal contribution.

** Corresponding author.

shown to still remain ill-suited for producing high-quality MIs [8, 23, 53, 54, 59], due to two specific requirements of MIs: *high-fidelity* and *SVG-format*.

Firstly, for **high-fidelity**, unlike common natural or animated image generation, which typically involves generative models hallucinating or inferring additional visual details to enhance perceptual plausibility and aesthetics [9, 59], MI generation aims to faithfully present the core ideas of a paper. This demands accurate visualization of concepts described in the paper (e.g., designed network modules), clear layout allocation, and precise rendering of symbolic and textual elements (e.g., concept names and equations), thereby imposing specific requirements on image fidelity that differ from those of conventional image generation tasks [8, 53]. Achieving such fidelity can be particularly challenging due to the inherent characteristics of MIs. Specifically, AI research papers typically involve multiple core concepts, each of which encodes complex semantics. To clearly convey these concepts, MIs often exhibit highly complex structures, usually comprising multiple visual modules that themselves contain rich internal substructures. Such intricate composition necessitates fine-grained and precise control over the generation process. Moreover, the research community evolves rapidly, and scientific papers frequently introduce new concepts. Such dynamics further increase the difficulty of accurately depicting core concepts in MIs and require the generation system to keep aligned with the rapidly evolving research community. Besides fidelity, MIs are increasingly expected in **Scalable Vector Graphics (SVG) format**, which preserves visual quality under arbitrary scaling and supports flexible post-editing, and this practice has become increasingly common in the research community. Yet, image-generation-based methods [11, 29, 33, 50, 57] usually produce resolution-limited and hard-to-edit raster images. This limitation is difficult to overcome through straightforward solutions such as converting raster outputs into SVGs via post-processing [7, 14, 58], which still inherit the fidelity limitations of the underlying generation models and even often introduce additional artifacts, leading to fragmented and hard-to-edit primitives [39, 52].

To this end, automating the creation of high-fidelity SVG-format MIs still remains a very challenging and largely underexplored problem. Nevertheless, we observe that human authors are still able to produce high-quality MIs, albeit with substantial time investment. This naturally motivates us to reflect on how human authors draw MIs. **Firstly**, although MIs exhibit intricate compositions and can vary substantially in overall appearance across different papers, they can still share similar visual components at a finer granularity. For example, distinct model architectures may include common components (e.g., encoder-decoder structures and attention blocks). We find some authors will store these common components from previously created MIs as templates (which we call *drawing middlewares* here) in their personal repositories. In this way, authors can construct complex new MIs through lightweight adjustments and assembly of these previously stored middlewares, thereby substantially simplifying and facilitating the challenging MI creation process. **Besides**, we also observe that authors often create new component middlewares as new concepts emerge and also continuously evolve (update) existing ones, to keep them aligned with the dynamically

evolving research community. **Moreover**, when handling complex MIs, instead of constructing the entire figure in one go, authors often draw in a trial-and-error manner, by tentatively adding a few strokes to see how the figure takes shape. They usually selectively proceed when the current attempt appears promising; otherwise, they revert to a previously satisfactory version and explore alternative drawing directions. **Lastly**, authors typically rely on professional drawing software to construct MIs, which usually supports exporting figures in SVG format.

Through the above drawing practices, human authors can well handle the inherent characteristics of MIs and effectively produce high-fidelity SVG-format MIs. Inspired by this, we propose **FigAgent**, the first multi-agent framework that automatically mirrors humans’ practices in creating, evolving, and orchestrating (assembling) reusable drawing middlewares for drawing high-fidelity SVG MIs of AI scientific papers. Specifically, our framework drives a vision-LLM-powered agent to automatically collect MIs from online published papers, including those from top-tier venues (e.g., ECCV and CVPR), to construct an *experience dataset*, from which it distills common visual components and encapsulates them into a **drawing middleware repository**. Notably, each middleware is implemented as *an executable Python function* corresponding to a DrawIO-supported drawing operation, parameterized by attributes such as position and size. For example, a middleware `Attention_map(x, y, w, h, "diagonal")` places a diagonal attention map at location (x, y) with width w and height h on the canvas. During MI generation, our framework employs another agent that interacts with professional drawing software (e.g., DrawIO) and autonomously orchestrates appropriate middlewares to draw complex MIs. Notably, since these middlewares are executable Python functions that encapsulate SVG code, MIs generated by the agent can be seamlessly rendered into SVG format and remain readily editable. Besides, inspired by Darwin’s theory of evolution and Genetic Programming [10, 17, 25, 55], our framework continuously evolves the middleware repository to improve its overall efficacy and keep it aligned with newly emerging concepts in the rapidly evolving research community.

More importantly, given the considerable complexity of MIs, which usually comprise multiple interrelated visual modules whose individual depiction and joint spatial arrangement critically affect the overall figure quality, our FigAgent further introduces **Explore-and-Select**, a novel drawing strategy that mirrors human authors’ trial-and-error drawing behavior to facilitate effective middleware orchestration on the canvas. Our Explore-and-Select strategy models the drawing process as a search tree [13, 35, 47] and guides the agent to explore proper middleware orchestration manners, encouraging the generation process to progress along promising paths.

Our contributions are summarized as follows. 1) Our FigAgent mimics the middleware creation, evolution, and orchestration practices of human authors, enabling the direct generation of high-fidelity SVG MIs for AI scientific papers. 2) By creating and evolving drawing middlewares through agent-driven collaboration, and by orchestrating them under the Explore-and-Select drawing strategy, our framework can effectively handle complex MIs while remaining

aligned with the rapidly evolving research community. 3) Our method achieves state-of-the-art performance on evaluated benchmarks.

2 Related Work

Recent advances in generative techniques [1, 27, 28] have demonstrated remarkable success across various domains [15, 18–21, 24, 44], also inspiring growing interest in automatic scientific illustration generation. Existing approaches [2–4, 8, 34, 40, 48, 50] can be categorized into two paradigms: (1) methods that rely on image generation models, and (2) methods that generate code that can be seamlessly converted into SVGs.

Some works [8, 33, 34, 50] utilize image-generation models (e.g., diffusion models) to produce illustrations for scientific papers. For instance, FigGen [33] and OCR-VQGAN [34] train text-to-image models to generate pixel-based illustrations from textual prompts, while AutoFigure [58] and PaperBanana [57] further employ LLMs to parse paper texts before invoking image generation models. To enhance editability, some works [5, 14] introduce post-processing techniques. EditBanana [5] uses OCR and segmentation to detect and convert textual and basic graphic elements into editable formats, while SuperSVG [14] adopts differentiable vectorization to convert raster images. Yet, such post-hoc approaches still inherit the fidelity limitations of the underlying image-generation models, and the vectorized outputs often exhibit imprecise geometric relationships and irregular paths [3, 42], limiting their suitability for high-quality MI generation.

Meanwhile, directly generating editable SVG illustrations usually better aligns with the needs of the research community. Along this line, some works [2–4, 40] leverage LLMs to generate $\text{\LaTeX}$ TikZ code for SVG-format illustrations, while others [32, 38, 41, 42, 46, 48] explore generating XML-based code that can be directly rendered into SVGs. Despite the progress, these methods usually remain limited to producing either simple icons [32, 38, 41, 42, 45, 48] or basic scientific charts [2, 4, 40] (e.g., bar plots), and thus still struggle with MI generation. A more recent attempt [12] explores generating code for MI layout, which, however, still relies on image generation models for rendering main visual modules, and thus the resulting illustrations remain limited to editing.

Differently, our FigAgent autonomously creates and evolves drawing middlewares to support rendering complex MIs, and introduces an Explore-and-Select strategy to guide the drawing process, which, to the best of our knowledge, for the first time enables the direct generation of high-fidelity SVG MIs.

3 Proposed Method

Given a paper text p (typically the text of the introduction and method sections), our goal is to automatically generate a high-quality MI (m) that intuitively conveys the core ideas of p . The MI m is stored as XML-based DrawIO code, which can be exported to SVG format for visualization and can be readily edited by authors. As shown in Fig. 1, our framework comprises two key

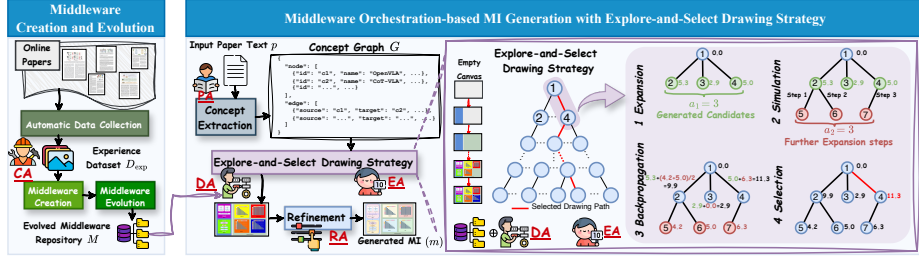


Fig. 1: Overview of our FigAgent. Our framework comprises five agents: CA, PA, DA, EA, and RA. In the *Middleware Creation and Evolution* stage, the CA creates a set of middlewares $\mathcal{M}$ from the automatically collected $\mathcal{D}_{\text{exp}}$, and evolves $\mathcal{M}$ to improve its efficacy and maintain alignment with the evolving research community. During *Middleware Orchestration-based MI Generation*, the PA parses paper text p into a concept graph $\mathcal{G}$. The DA then orchestrates middlewares from $\mathcal{M}$ to gradually render concepts onto the canvas, under the Explore-and-Select strategy (we show a toy case where (a_1, a_2, β) is simply set to $(3, 3, 1)$ for ease of understanding), with the EA providing evaluation feedback. Lastly, the RA refines the result to produce the MI m .

stages, both executed through the collaboration of LLM agents: **1) Middleware Creation and Evolution.** Our framework automatically and continuously collects publicly available AI scientific papers and constructs an *experience dataset* $\mathcal{D}_{\text{exp}} = \{(p_n, m_n)\}_{n=1}^{N_{\text{exp}}}$, where N_{exp} denotes the number of paper-MI pairs. Through multi-agent collaboration, reusable common visual components are identified and extracted from MIs in $\mathcal{D}_{\text{exp}}$, and then encapsulated into callable Python functions to construct a drawing middleware repository $\mathcal{M}$ via our dedicated middleware creation and evolution mechanisms, which can support subsequent MI generation. **2) Middleware Orchestration-based MI Generation with Explore-and-Select.** Equipped with $\mathcal{M}$, our framework parses the paper text p to extract key concepts and orchestrates appropriate middlewares to render these concepts. This process is guided by the *Explore-and-Select* drawing strategy, under which our agents collaboratively perform drawing, evaluation, and refinement to generate high-quality MIs.

Below, we first present the agents in our framework in Sec. 3.1, and then describe how $\mathcal{M}$ is created and evolved in Sec. 3.2. Finally, we discuss the overall MI generation process guided by the Explore-and-Select strategy in Sec. 3.3.

3.1 Agents in FigAgent

We begin by analyzing the human drawing practices to inform the design of our framework. Before drawing, authors often accumulate drawing experiences by studying high-quality MIs from existing papers, and some authors even distill common visual components into reusable drawing templates (middlewares) to subsequently support the drawing of new MIs. When drawing for a given paper text, they usually first parse the text to identify key concepts and their relation-

ships, then progressively render each concept onto the canvas through iterative trial-and-error revisions, and finally perform a refinement step to further adjust the overall MI, e.g., adding connections and tuning alignment. Motivated by this analysis, as shown in Fig. 1, our FigAgent framework mainly consists of the following five components (agents) that mirror humans’ drawing workflow, *with more details (including LLM prompts) provided in Supplementary*.

(1) Constructor Agent (CA): The vision-LLM-powered CA automatically scrapes publicly available paper texts and corresponding MIs, and filters out low-quality paper-MI pairs to construct the experience dataset $\mathcal{D}_{\text{exp}}$. Based on $\mathcal{D}_{\text{exp}}$, the CA then constructs and maintains a drawing middleware repository $\mathcal{M}$ to support MI generation. **(2) Parser Agent (PA):** Given a paper text p , the LLM-powered PA first identifies the paper’s research theme t , then extracts core concepts and organizes them into a concept graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$, where $\mathcal{C} = \{c_i\}$ is the set of concept nodes and $\mathcal{E}$ encodes inter-concept relationships. **(3) Drawer Agent (DA):** The vision-LLM-powered DA interfaces with DrawIO and draws the concepts in $\mathcal{G}$ onto the canvas one by one. Notably, during this process, the DA will select appropriate middlewares from $\mathcal{M}$ and orchestrate them for drawing. **(4) Evaluator Agent (EA):** Powered by a vision-LLM, the EA assesses MI quality, producing textual revision feedback and a numeric quality score that serves as the reward signal in our Explore-and-Select strategy. **(5) Refiner Agent (RA):** Powered by a vision-LLM, the RA applies final adjustments to MIs, including adding inter-concept connections based on $\mathcal{G}$, tuning spacing, and harmonizing the overall visual style.

These agents collaboratively complete the two key stages of our framework for high-quality MI generation. Specifically, the CA, PA, DA, and EA participate in the creation and evolution process of the *Drawing Middleware Repository*, while, guided by our Explore-and-Select drawing strategy, the PA, DA, EA, and RA collectively accomplish *Middleware Orchestration-based MI Generation*.

3.2 Middleware Creation and Evolution

Given the inherent complexity of MIs, faithfully drawing them from paper texts while ensuring convenient editability remains challenging for the DA. Nevertheless, we observe that although different MIs usually vary substantially in overall appearance, they can still share similar visual components at a finer granularity, presenting a key opportunity to mitigate this challenge.

Based on this, we propose a novel *middleware-driven* MI generation paradigm. Our framework drives the CA to identify recurring visual components across MIs and encapsulate them into a drawing middleware repository $\mathcal{M}$, so that complex MI generation can be decomposed into the *selection*, *parameterization*, and *orchestration* of well-established middlewares, greatly reducing drawing difficulty. Besides, leveraging the dual nature of SVG-format MIs, which can be perceived as both visual graphics and XML-based code, each middleware is implemented as an executable Python function [25] that generates corresponding code for a reusable visual component. This mechanism simultaneously ensures visual fidelity and convenient editability and is naturally compatible with vision-LLM-

powered agents that exhibit strong capabilities in both visual understanding and code generation [16,22,43,49,51]. Moreover, our framework also drives the CA to continuously evolve $\mathcal{M}$ through LLM-simulated biological genetics, maintaining alignment with newly emerging concepts (e.g., novel network modules) in the rapidly evolving research community while enhancing overall effectiveness.

Below, we first describe how our framework initially creates $\mathcal{M}$, and then elaborate on the middleware evolution mechanism that makes $\mathcal{M}$ ready for use.

Middleware Creation. To create $\mathcal{M}$, the CA first automatically collects publicly available paper-MI pairs to construct the experience dataset $\mathcal{D}_{\text{exp}}$. By analyzing $\mathcal{D}_{\text{exp}}$, we obtain two key observations that inform the construction of $\mathcal{M}$: (1) MI presentation is closely related to the paper’s research theme (topic), and even the same concept can be visualized differently across themes. For example, a “feature map” is often depicted as a diagonal grid pattern in LLM reasoning papers, whereas in object detection papers, it may appear as a multi-scale feature pyramid. (2) Within the same research theme, similar concepts often tend to exhibit common visual components. Given this, our framework organizes $\mathcal{M}$ in a three-level structure [25]: *theme* (e.g., object detection, image generation), *concept* (e.g., Transformer), and *component* (e.g., attention layers), to facilitate the DA in retrieving appropriate middlewares according to drawing requirements (detailed in Sec. 3.3). Based on this structure, our framework can create middlewares at the component level for each theme–concept pair.

For each paper-MI pair (p, m) in $\mathcal{D}_{\text{exp}}$, the PA first parses p to obtain a set of core concepts $\mathcal{C}$ and identifies the paper theme t . Then, for each concept c in $\mathcal{C}$, the CA jointly analyzes both the visual SVG and XML-based DrawIO code of the MI m , with a focus on the regions relevant to c , and identifies a set of key components that constitute concept c (detailed prompt is in Supplementary). More crucially, instead of storing these identified components as static code segments, the CA abstracts each into a parameterized middleware: it encapsulates the corresponding code segment as a callable Python function, while exposing key drawing attributes (e.g., position, size, and visual patterns) as adjustable function parameters with sensible defaults inferred from the original MI. For example, a feature pyramid component is abstracted into a middleware `Feature_Pyramid(x, y, w, h, num_levels, shape_mode)`, where `x`, `y`, `w`, `h` control its placement and size on the canvas, `num_levels` specifies the number of pyramid layers, and `shape_mode` (e.g., “cuboid”, “rectangle”) specifies the geometric form used to depict each pyramid level. In this way, a single middleware can flexibly adapt to diverse drawing contexts through proper parameter configuration. The resulting function-based middlewares are then indexed under the theme–concept pair (t, c) and added into the initial middleware repository $\mathcal{M}$.

By repeating the above middleware creation process for all samples in $\mathcal{D}_{\text{exp}}$, our framework creates the initial middleware repository $\mathcal{M}$ defined as

$$\mathcal{M} = \bigcup_{(t_i, c_j) \in T \times C} \{(t_i, c_j, F_{ij})\}, \quad (1)$$

where $T = \{t_i\}_{i=1}^{N_T}$ and $C = \{c_j\}_{j=1}^{N_C}$ are sets of *themes* and *concepts*, and $F_{ij} = \{f_1, f_2, \dots, f_{N_{ij}}\}$ denotes the set of middlewares associated with theme t_i and concept c_j . Yet, since middlewares are independently extracted from different papers, the initially created $\mathcal{M}$ may contain semantically overlapping concepts (e.g., “self-attention module” and “multi-head attention”) under the same research theme, introducing redundancy. To obtain a more compact repository, the CA further performs a *merging* step: it computes semantic similarity among concepts within each theme, consolidates highly similar concepts into unified entries, and leverages the LLM to identify and remove functionally redundant middlewares among the merged concepts (more details in Supplementary). The resulting consolidated $\mathcal{M}$ is then passed to the middleware evolution stage.

Middleware Evolution. Due to the heterogeneous quality of MIs in $\mathcal{D}_{\text{exp}}$, the initially created $\mathcal{M}$ may still contain some low-quality middlewares, necessitating deletion or refinement. Moreover, exploring and incorporating new middlewares into $\mathcal{M}$ is also essential to adapt to newly emerging concepts (e.g., novel model architectures) as the research community continues to evolve. To this end, our framework drives the CA to periodically incorporate paper-MI samples from newly published papers into $\mathcal{D}_{\text{exp}}$, and, from these newly added samples, re-invoke the aforementioned middleware creation mechanism to create new middlewares and integrate them into $\mathcal{M}$. Moreover, our framework also seeks to evolve the middlewares in $\mathcal{M}$ to enhance their overall efficacy. Yet, this is non-trivial, since unlike vectorized parameters in deep learning models, middlewares are implemented as Python functions and thus are not directly differentiable.

Drawing inspiration from evolutionary biology and genetic programming [10, 17, 56], especially in the evolution of Python functions [25], our framework employs an LLM-driven middleware evolution mechanism to optimize $\mathcal{M}$ in a non-gradient manner. Specifically, our framework conceptualizes each middleware f_k in $\mathcal{M}$ as an individual, the initially created repository $\mathcal{M}$ as the population, and the experience dataset $\mathcal{D}_{\text{exp}}$ as the environment. The LLM-powered CA is then prompted to simulate a biological evolutionary process over this population, aiming to improve its *fitness* to the environment, namely, the overall middleware efficacy. This process proceeds iteratively through two alternating steps: *Middleware Evaluation*, which assesses the population’s fitness, and *Middleware Update*, which emulates biological evolutionary operators, including selection, crossover, and mutation, to optimize the repository, with the former guiding the latter and controlling the iterative process.

Middleware Evaluation. The middleware evaluation step measures the fitness of the population $\mathcal{M}$ within the environment $\mathcal{D}_{\text{exp}}$, serving as the objective function that informs the consequence of each middleware update step: guiding the framework to retain beneficial updates and discard detrimental ones. A straightforward approach for measuring fitness is to evaluate the MI generation performance of our framework using $\mathcal{M}$. However, solely considering this can be one-sided, as $\mathcal{M}$ may contain low-quality middlewares that are rarely invoked during generation and thus have a negligible impact on the final MI quality, yet their presence can still mislead subsequent middleware update operations

(e.g., being selected as parents for crossover). Our framework thus additionally evaluates each individual middleware’s efficacy, guiding the update process to optimize the repository $\mathcal{M}$ at a finer granularity. During MI generation, the DA selects and orchestrates middlewares to render concepts one by one (detailed in Sec. 3.3). This process naturally provides a signal for evaluating individual middleware efficacy: each time a middleware f_k is used, the EA assesses the visualization quality of the rendered concept it contributes to (detailed in Supplementary). Our framework tracks the utilization of each middleware f_k and defines the *Middleware Efficacy Score* (MES) as $\text{MES}_{f_k} = S_{f_k}/N_{f_k}$, where S_{f_k} is the sum of quality scores across all concept renderings that f_k contributes to and N_{f_k} is its total number of utilizations. A higher MES reflects greater efficacy. Based on this, we define the objective function that jointly incorporates generation-level and individual-level metrics:

$$\mathcal{L} = \frac{1}{|\mathcal{P}|} \sum_{p_{\text{exp}} \in \mathcal{P}} q(p_{\text{exp}}) + \frac{1}{|\mathcal{M}_{\mathcal{P}}|} \sum_{f_k \in \mathcal{M}_{\mathcal{P}}} \text{MES}_{f_k}, \quad (2)$$

where $\mathcal{P}$ denotes a batch of paper texts sampled from $\mathcal{D}_{\text{exp}}$, $q(p_{\text{exp}})$ is the quality score assessed by the EA for the MI generated from p_{exp} , and $\mathcal{M}_{\mathcal{P}} \subseteq \mathcal{M}$ is the subset of middlewares invoked during MI generation for $\mathcal{P}$. By incorporating both generation-level and individual-level evaluation, this objective function provides comprehensive guidance for the subsequent middleware update step.

Middleware Update. In this step, our framework employs the CA to optimize $\mathcal{M}$ through three key operations analogous to biological evolution [25]: selection, mutation, and crossover. Each operation is realized via carefully crafted prompts within the CA (provided in Supplementary), enabling it to refine or delete existing middlewares, and explore new ones. *(1) Selection.* Mimicking natural selection, where only the fittest individuals survive, the CA retains effective middlewares and removes ineffective ones based on their MES, thereby eliminating noise and preventing low-quality middlewares from propagating into subsequent operations. *(2) Mutation.* Analogous to genetic mutation that introduces phenotypic diversity within a species, the CA modifies a middleware function by adjusting its parameters or expanding its functionality, thereby inducing variations that encourage the exploration of potentially superior middleware designs. *(3) Crossover.* Analogous to genetic recombination that produces offspring inheriting advantageous traits from both parents, the CA generates new middlewares by combining complementary features from two or more selected middlewares, creating potentially more versatile and effective variants.

Overall, the middleware evolution process is guided by the objective function $\mathcal{L}$ in Eq. (2), which is maximized to improve the overall efficacy of the repository. Middleware updates that reduce $\mathcal{L}$ are discarded via a rollback mechanism, and early-stopping is applied when no significant improvement is observed over consecutive iterations. The evolution process terminates once the stopping condition is met or the maximum number of iterations is reached. We provide a visualization of an example middleware’s dynamics during the middleware creation and evolution processes in Fig. 2.

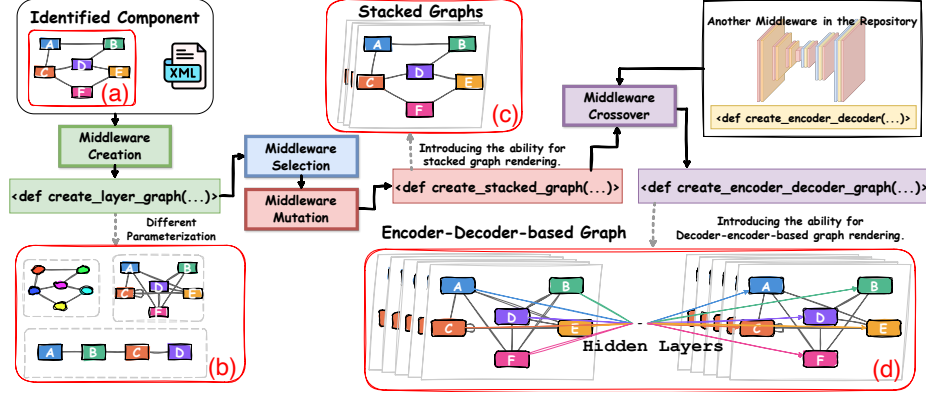


Fig. 2: Visualization of an example middleware’s dynamics during the middleware creation and evolution processes. We take the middleware `create_layer_graph` in our repository as an example. The red box (a) visualizes the identified visual component from which `create_layer_graph` is created; (b) shows outputs rendered by this middleware with different parameters; (c) illustrates the output rendered by its mutated version (`create_stacked_graph`); and (d) shows the output rendered by its crossover version (`create_encoder_decoder`). We also provide the corresponding Python functions for the initially created, mutated, and crossover middlewares in Supplementary.

3.3 Middleware Orchestration-based MI Generation with Explore-and-Select

After creating and evolving the middleware repository $\mathcal{M}$ and equipping it for the DA, our framework should be ready for MI generation. A natural pipeline thus is to parse the paper text to extract key concepts via the PA, render them via the DA, and finally employ the RA to refine the resulting MI. Besides, to further simplify complex MI generation and improve the quality of generated MIs, our framework mirrors the common practice of human authors, who usually draw MIs progressively rather than construct the entire figure in one go. Based on this intuition, our framework drives the DA to render one concept at a time on the canvas, gradually completing the entire MI. We detail how the DA orchestrates middlewares from $\mathcal{M}$ to render each concept later.

However, even with $\mathcal{M}$ and progressive drawing, we observe that this pipeline can still yield sub-optimal MIs. This is mainly because the LLM-powered DA is inherently stochastic: when rendering the same concept, it can produce substantially different visual outcomes by choosing different middlewares or even applying the same middlewares with different parameterization. We refer to these variations as distinct *drawing choices*. More importantly, since an MI comprises multiple interrelated concepts whose individual depiction and joint spatial arrangement critically determine the overall figure quality, drawing choices are path-dependent: the choice made for one concept constrains how subsequent

concepts can be depicted and arranged. Thus, inappropriate early choices can propagate through the rendering sequence and gradually degrade the final MI.

To tackle this, our framework employs a Monte Carlo Tree Search [6, 26, 35] inspired drawing strategy, namely **Explore-and-Select**, which first *explores* multiple candidate drawing choices and evaluates their long-term effects on the MI, and then *selects* the most promising one to advance the MI, thereby alleviating premature commitment to suboptimal choices and enabling the discovery of more coherent and synergistic combinations of drawing choices.

Below, we first detail how the DA selects and orchestrates middlewares to render each concept, and then elaborate on how the Explore-and-Select strategy guides concept rendering towards promising paths.

Concept Rendering. During MI generation, the DA selects and orchestrates appropriate middlewares from $\mathcal{M}$ to sequentially render the concepts $\mathcal{C} = \{c_i\}$ extracted from the paper text p . This process can be formulated as:

$$m^{(i)} = \text{Drawer}(t, c_i, m^{i-1}; \mathcal{M}, H), \quad (3)$$

where $m^{(i)}$ denotes the generated code segment for rendering concept c_i , m^{i-1} is the accumulated code after the first $i-1$ concepts have been rendered, t is the research theme inferred from p , and H denotes the historical middleware usage experiences. Concretely, this rendering process involves two steps: *retrieval* and *filtering*. In *retrieval*, our framework computes the semantic embedding similarity between the target concept c_i and concepts under the same theme t in $\mathcal{M}$, and collects middlewares associated with the top- K most similar concepts to form a candidate set F_{c_i} . Since such retrieval may introduce irrelevant or redundant middlewares due to coarse semantic embedding matching, a subsequent *filtering* step prompts an LLM to prune F_{c_i} , and retain only the most suitable middlewares (detailed prompt in Supplementary). The resulting filtered set, denoted as F'_{c_i} , together with H , is then provided to the DA for concept rendering. More details about the concept rendering process are in Supplementary.

Explore-and-Select Strategy. To guide the DA to select promising drawing choices, as shown in Fig. 1, the Explore-and-Select strategy formulates the MI generation process as a search tree [35]. Each node represents an intermediate canvas state (i.e., a partially constructed MI), with the root node denoting an empty canvas. Each edge denotes a drawing choice made by the DA that advances the MI by drawing the next concept through the aforementioned concept rendering process. This strategy iteratively builds a promising path through the tree. At each iteration, it first *expands* the current node by exploring multiple candidate drawing choices, producing a set of child nodes, then *simulates* further expansions from these child nodes to score the long-term impact of these drawing choices, *backpropagates* the resulting scores to update their evaluations, and finally *selects* the most promising child node to advance the MI. This four-step cycle repeats until all concepts are completed. We detail each step below.

Expansion. Starting from the terminal node of the current path (node 1 in Fig. 1), or the root node (empty canvas) in the first iteration, the DA renders the next concept using a_1 different drawing choices, producing a_1 candidate MI

states (child nodes; i.e., nodes 2-4 in Fig. 1). These child nodes are then scored by the EA and ranked. Since each child node extends its parent node by completing one additional concept, we define the *node score* as the incremental gain in MI quality score relative to its parent, rather than the absolute score assessed by the EA. Accordingly, the root node (empty canvas) is assigned a node score of zero. Notably, if all child nodes receive node scores below a predefined threshold, the DA regenerates a_1 children by incorporating the EA’s revision feedback.

Simulation. While the above node scores can reflect the immediate impact of drawing choices, they still fail to capture their effects on future compositions. To this end, we simulate further a_2 expansions from these a_1 generated nodes to estimate the long-term impacts of drawing choices. At each expansion step, to prevent the agent from being trapped in local optima by only selecting the currently best-performing nodes, we also retain exploration opportunities for choices that exhibit lower initial scores but may still possess latent potential. Specifically, a UCT-inspired criterion [35] is used to balance exploitation (favoring nodes with higher estimated values) and exploration (encouraging less-explored nodes) by computing the *UCT value* for the nodes: $UCT_j = \frac{2.0}{\text{rank}_j + 1} + \beta \sqrt{\frac{2.0}{\text{child_count}_j + 1}}$, where j indexes a node, rank_j denotes its relative rank based on the initial node scores, child_count_j is the number of child nodes already expanded from node j , and β controls the exploration-exploitation trade-off. At each step, the DA advances the node with the highest UCT value by rendering one subsequent concept, generating a new child node.

Backpropagation. To incorporate the long-term estimates obtained during simulation into the current decision, the scores of newly generated child nodes (nodes 5, 6, 7 in Fig. 1) are backpropagated through the tree. Concretely, a parent node’s score is updated by adding the average score of its child nodes. In this way, each drawing choice is assessed not only by its immediate impact on the MI but also by its long-term effects on possible future drawing paths.

Selection. The node with the highest updated score (node 4 in Fig. 1) is selected as the new terminal node for the next iteration.

The above four-step cycle will repeat until all concepts are completed. We provide more details about the Explore-and-Select strategy in Supplementary.

3.4 Preparation and Testing.

Preparation. Prior to MI generation, the CA automatically constructs the middleware repository $\mathcal{M}$ from the collected experience dataset $\mathcal{D}_{\text{exp}}$ as detailed in Sec. 3.2. By automatically and periodically incorporating newly published papers and invoking the middleware creation and evolution mechanisms, the CA keeps $\mathcal{M}$ ready for use and aligned with the evolving research community. **Testing.** Given a paper text p , the PA first parses it into a concept graph $\mathcal{G}$. The DA then renders concepts in $\mathcal{G}$ by orchestrating middlewares from $\mathcal{M}$ under the Explore-and-Select strategy, with the EA providing evaluative guidance during this process. Finally, the RA performs refinement to obtain the final MI m .

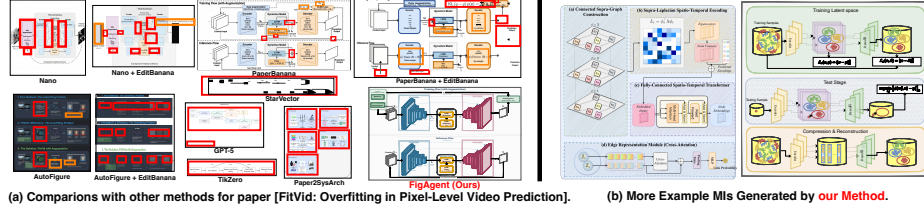


Fig. 3: Qualitative Results. **Red boxes** denote structural defects, e.g., layout inconsistencies and missing components. **Orange boxes** denote low-fidelity details like blurriness. Zoom in for better view. **More zoomed-in examples are in Supplementary.**

4 Experiments

Implementation Details. We use GPT-5 [36] to power our agents, and use `all-mpnet-base`

`-v2` [31] to encode concepts into embeddings for middleware retrieval. More implementation details (including agent prompts) are in Supplementary.

Dataset. Following the data collection pipeline of [23], our framework collects MIs and corresponding paper texts (from major venues such as ECCV and CVPR), and extracts the XML code from the MI files, yielding 4,964 paper-MI (XML code) pairs in total. We randomly sample 80% of the pairs to build $\mathcal{D}_{\text{exp}}$, while the remaining 20% serve as the test set (named **FigAgentBench**) for evaluation. We also conduct evaluation on PaperBananaBench [57]. More dataset details are in Supplementary.

Evaluation Metrics. We use DiagramEval [23], a metric designed for assessing AI-generated MIs, which conceptualizes the generated MI and its corresponding paper text as graphs, and evaluates their node alignment (**NA**) and path alignment (**PA**). We compute the NA and PA scores for each generated MI, and report the average scores across all MIs in FigAgentBench and PaperBananaBench, respectively. Besides, we use a Vision-LLM evaluator [57, 58] as a complementary metric that assesses MI quality and reports an overall quality score, denoted as **VLM**. More details about evaluation metrics are in Supplementary.

4.1 Evaluation on MI Generation

Baseline Methods. We compare our framework against two categories of baselines. (1) *Image-based Methods*, which employ image generation models as the core component for MI generation. We consider Nano Banana Pro (Nano) [11], which is prompted to produce raster MIs in an end-to-end manner. This category also includes AutoFigure [58] and PaperBanana [57], which additionally leverage LLMs to parse paper texts before invoking image generation models. (2) *SVG-based Methods*, which can directly generate easily editable SVG illustrations. We consider GPT-5 [36], which is prompted to generate XML-based DrawIO code from paper texts, as a representative baseline. We also include TikZero [2] and

StarVector [32], which fine-tune LLMs to generate TikZ or XML code for producing SVGs, as well as Paper2SysArch [12], which employs agents to generate XML code. Besides, we also utilize EditBanana [5], a widely adopted raster-to-SVG tool, to transform the outputs of image-based methods into editable SVGs for further comparison. More details about these baselines are in Supplementary.

Table 1: Quantitative comparisons of different methods on MI generation quality.

Method	FigAgentBench							PaperBananaBench [57]						
	NA↑			PA↑			VLM↑	NA↑			PA↑			VLM↑
	Prec.	Recal.	F1	Prec.	Recal.	F1		Prec.	Recal.	F1	Prec.	Recal.	F1	
Image-based Methods														
Nano [11]	35.1	40.0	36.9	22.3	14.3	16.1	55.7	38.3	39.2	38.1	22.8	17.9	18.2	59.2
PaperBanana [57]	46.7	47.1	46.3	37.7	22.8	26.9	60.3	52.6	44.0	47.2	39.1	22.2	28.7	66.6
AutoFigure [58]	32.8	37.4	34.5	34.3	20.9	23.9	53.1	42.4	40.3	39.9	38.6	23.5	27.1	64.2
SVG-based Methods														
GPT-5 [36]	35.6	21.6	25.6	27.3	10.1	13.8	31.1	39.4	36.3	37.5	23.5	11.2	14.4	36.2
TikZero [2]	20.4	11.2	14.9	9.6	1.7	2.3	4.2	23.3	9.4	13.2	13.2	3.0	3.2	8.4
StarVector [32]	14.7	10.5	11.4	8.1	2.0	2.7	4.5	16.6	8.8	10.7	11.8	2.4	3.1	7.2
Paper2SysArch [12]	42.6	30.9	34.8	26.1	18.8	20.3	35.2	39.1	37.5	36.3	22.5	19.7	19.4	39.3
Nano+EditBanana [11]	34.6	36.1	35.3	18.4	11.9	12.6	50.4	37.2	37.0	35.7	17.6	13.5	14.3	56.8
PaperBanana+EditBanana [57]	44.9	43.6	44.5	31.0	18.4	21.2	55.9	50.9	39.6	43.8	31.6	17.5	20.8	58.7
AutoFigure+EditBanana [58]	32.0	34.4	33.2	27.0	18.7	21.3	51.6	41.0	37.4	38.0	34.3	20.2	24.0	54.3
FigAgent (Ours)	53.7	51.9	51.4	40.9	28.5	32.6	61.4	54.8	50.5	51.6	44.1	33.7	36.7	68.9

Results. Tab. 1 shows the comparative results. Among image-based methods, by using LLMs to parse paper texts, PaperBanana and AutoFigure achieve better performance compared to the simpler Nano baseline. However, these methods still suffer from low PA scores due to the inherent fidelity limitations of the underlying image generation models, which frequently produce visually appealing but structurally incorrect outputs. On the other hand, we find that SVG-based methods such as StarVector and TikZero, despite producing editable SVG outputs, still underperform image-based methods across most metrics. This is mainly because these methods are specially designed for icons and simple data charts, and still struggle to adapt to more complex MI generation, which requires accurately depicting intricate visual components and their compositional relationships. We also find that Paper2SysArch, although achieving relatively good results on layout-related metrics (NA and PA), still exhibits notable deficiencies in the visual quality (as reflected by VLM Score).

Differently, by constructing drawing middlewares to support complex concept rendering and by guiding MI generation through the Explore-and-Select strategy that evaluates long-term impacts of drawing choices, our framework achieves the best performance across all metrics. Qualitative comparisons are shown in Fig. 3.

4.2 Framework Analysis

Below, we conduct additional experiments on the FigAgentBench to further analyze the key characteristics of our framework. More results, including human evaluation, are in Supplementary.

Impact of Main Mechanisms of FigAgent. We test the following variants: 1) **w/o middleware**, where the DA directly generates XML code segment

for rendering each concept without using middleware; **2) w/o Explore-and-Select**, where we remove our Explore-and-Select strategy, i.e., the DA renders the concepts in $\mathcal{G}$ one by one in a single pass, without trial-and-error drawing. As shown in Tab. 2, both the middleware repository and the Explore-and-Select strategy contribute significantly to overall efficacy.

Table 2: Evaluation on main mechanisms.

Method	NA $\uparrow$			PA $\uparrow$			VLM $\uparrow$
	<i>Prec.</i>	<i>Recal.</i>	<i>F1</i>	<i>Prec.</i>	<i>Recal.</i>	<i>F1</i>	
w/o middleware	29.8	24.3	26.5	24.7	18.3	19.9	39.5
w/o Explore-and-Select	39.3	31.5	33.6	32.4	20.2	23.8	48.2
Ours	53.7	51.9	51.4	40.9	28.5	32.6	61.4

Impact of Middleware Evolution Mechanism. We compare the following variants: **1) w/o evolution**, where the middleware evolution process is entirely removed, and the DA can only invoke middlewares from the initially created repository for drawing MIs; **2) w/o selection**, **3) w/o mutation**, and **4) w/o crossover**, where the corresponding operation in the evolution process is removed. As shown in Tab. 3, our full framework outperforms all ablated variants. Specifically, selection *removes* low-quality middlewares, mutation *refines* existing middlewares, and crossover *explores new* middleware designs, all of which contribute positively to the efficacy of middleware evolution.

Table 3: Evaluation on middleware evolution mechanism.

Method	NA $\uparrow$			PA $\uparrow$			VLM $\uparrow$
	<i>Prec.</i>	<i>Recal.</i>	<i>F1</i>	<i>Prec.</i>	<i>Recal.</i>	<i>F1</i>	
w/o evolution	37.3	27.2	31.8	30.2	19.7	22.9	45.5
w/o selection	40.1	30.7	33.9	32.1	20.3	24.2	47.6
w/o mutation	48.9	46.6	46.8	36.2	27.1	29.5	59.3
w/o crossover	46.5	44.3	43.8	35.8	25.8	28.3	54.2
Ours	53.7	51.9	51.4	40.9	28.5	32.6	61.4

5 Conclusion

We propose FigAgent, a novel multi-agent framework for MI generation. By distilling drawing experiences from published scientific papers to create and evolve a drawing middleware repository, and by performing drawing under an MCTS-driven Explore-and-Select strategy that evaluates the long-term impact of each drawing choice, our framework automatically generates high-fidelity SVG MIs for scientific papers. Extensive experiments demonstrate the efficacy of our method.

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. Belouadi, J., Ilg, E., Keuper, M., Tanaka, H., Utiyama, M., Dabre, R., Eger, S., Ponzetto, S.: Tikzero: Zero-shot text-guided graphics program synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17793–17806 (2025)
3. Belouadi, J., Lauscher, A., Eger, S.: Automatikz: Text-guided synthesis of scientific vector graphics with tikz. In: The Twelfth International Conference on Learning Representations (2024)
4. Belouadi, J., Ponzetto, S., Eger, S.: Detikzify: Synthesizing graphics programs for scientific figures and sketches with tikz. *Advances in Neural Information Processing Systems* **37**, 85074–85108 (2024)
5. BIT-DataLab: Edit-banana. <https://github.com/BIT-DataLab/Edit-Banana> (2024), gitHub repository, accessed 30 June 2026
6. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* **4**(1), 1–43 (2012)
7. Carlier, A., Danelljan, M., Alahi, A., Timofte, R.: Deepsvg: A hierarchical generative network for vector graphics animation. *Advances in Neural Information Processing Systems* **33**, 16351–16361 (2020)
8. Chang, Y., Feng, Y., Sun, J., Ai, J., Li, C., Zhou, S.K., Zhang, K.: Sridbench: Benchmark of scientific research illustration drawing of image generation model. arXiv preprint arXiv:2505.22126 (2025)
9. Chen, Z., Min, Y., Zhang, J., Yan, B., Wang, J., Wang, X., Shan, S.: A survey of multimodal hallucination evaluation and detection. arXiv preprint arXiv:2507.19024 (2025)
10. Fisher, R.A.: The genetical theory of natural selection: a complete variorum edition. Oxford University Press (1999)
11. Google DeepMind: Gemini 3 pro image model card (2025), <https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Image-Model-Card.pdf>, accessed 30 June 2026
12. Guo, Z., Liu, Z., Zhang, W.: Paper2sysarch: Structure-constrained system architecture generation from scientific papers. arXiv preprint arXiv:2511.18036 (2025)
13. Hart, P.E., Nilsson, N.J., Raphael, B.: A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* **4**(2), 100–107 (1968)
14. Hu, T., Yi, R., Qian, B., Zhang, J., Rosin, P.L., Lai, Y.K.: Supersvg: Superpixel-based scalable vector graphics synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 24892–24901 (2024)
15. Hui, X., Wu, Q., Rahmani, H., Liu, J.: Class-agnostic object counting with text-to-image diffusion model. In: European Conference on Computer Vision. pp. 1–18. Springer (2024)
16. Hurst, A., Lerer, A., Goucher, A.P., Perelman, A., Ramesh, A., Clark, A., Ostrow, A., Welihinda, A., Hayes, A., Radford, A., et al.: Gpt-4o system card. arXiv preprint arXiv:2410.21276 (2024)

17. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. *Statistics and computing* **4**(2), 87–112 (1994)
18. Li, Z., Qu, H., Kuen, J., Gu, J., Ke, Q., Liu, J., Rahmani, H.: Diffip: Representation fingerprints for robust ip protection of diffusion models. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 17035–17045 (2025)
19. Li, Z., Rahmani, H., Ke, Q., Liu, J.: Longdiff: Training-free long video generation in one go. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 17789–17798 (2025)
20. Li, Z., Rahmani, H., Zhang, J., Xue, Y., Mirmehdi, M., Kuen, J., Gu, J., Liu, J.: Diffgraph: An automated agent-driven model merging framework for in-the-wild text-to-image generation. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 36713–36723 (2026)
21. Li, Z., Wang, Y.: Better integrating vision and semantics for improving few-shot classification. In: *Proceedings of the 31st ACM international conference on multimedia*. pp. 4737–4746 (2023)
22. Li, Z., Wang, Y., Li, K.: Fewvs: A vision-semantics integration framework for few-shot image classification. In: *Proceedings of the 32nd ACM International Conference on Multimedia*. pp. 1341–1350 (2024)
23. Liang, C., You, J.: Diagrameval: Evaluating llm-generated diagrams via graphs. In: *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2025)
24. Liu, K., Mao, W., Shen, S., Jiao, X., Sun, Z., Chen, H., Shen, C.: Floating anchor diffusion model for multi-motif scaffolding. In: *Proceedings of the 41st International Conference on Machine Learning*. pp. 31691–31708 (2024)
25. Ma, Z., Huang, Z., Liu, J., Wang, M., Zhao, H., Li, X.: Automated creation of reusable and diverse toolsets for enhancing llm reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence* **39**(23), 24821–24830 (2025)
26. Marsland, T.A.: A review of game-tree pruning. *ICGA journal* **9**(1), 3–19 (1986)
27. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., Gao, J.: Large language models: A survey. *arXiv preprint arXiv:2402.06196* (2024)
28. Nichol, A.Q., Dhariwal, P.: Improved denoising diffusion probabilistic models. In: *International conference on machine learning*. pp. 8162–8171. PMLR (2021)
29. OpenAI: Dall-e 3 system card (2023), <https://openai.com/index/dall-e-3/>, accessed 30 June 2026
30. OpenAI: Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024)
31. Reimers, N., Team, S.T.: all-mpnet-base-v2. <https://huggingface.co/sentence-transformers/all-mpnet-base-v2> (2021), model card, accessed 30 June 2026
32. Rodriguez, J.A., Puri, A., Agarwal, S., Laradji, I.H., Rodriguez, P., Rajeswar, S., Vazquez, D., Pal, C., Pedersoli, M.: Starvector: Generating scalable vector graphics code from images and text. In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. pp. 16175–16186 (2025)
33. Rodriguez, J.A., Vazquez, D., Laradji, I., Pedersoli, M., Rodriguez, P.: Figgen: Text to scientific figure generation. *arXiv preprint arXiv:2306.00800* (2023)
34. Rodriguez, J.A., Vazquez, D., Laradji, I., Pedersoli, M., Rodriguez, P.: Ocr-vqgan: Taming text-within-image generation. In: *Proceedings of the IEEE/CVF winter conference on applications of computer vision*. pp. 3689–3698 (2023)
35. Shi, H., Li, Y., Chen, X., Wang, L., Hu, B., Zhang, M.: Animaker: Multi-agent animated storytelling with mcts-driven clip generation. In: *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*. pp. 1–11 (2025)

36. Singh, A., Fry, A., Perelman, A., Tart, A., Ganesh, A., El-Kishky, A., McLaughlin, A., Low, A., Ostrow, A., Ananthram, A., et al.: OpenAI GPT-5 System Card. arXiv preprint arXiv:2601.03267
37. Song, J., Meng, C., Ermon, S.: Denoising diffusion implicit models. In: International Conference on Learning Representations (2020)
38. Wang, F., Zhao, Z., Liu, Y., Zhang, D., Gao, J., Sun, H., Li, X.: Svgen: Interpretable vector graphics generation with large language models. In: Proceedings of the 33rd ACM International Conference on Multimedia. pp. 9608–9617 (2025)
39. Wang, Z., Huang, J., Sun, Z., Gong, Y., Cohen-Or, D., Lu, M.: Layered image vectorization via semantic simplification. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 7728–7738 (2025)
40. Wei, J., Tan, C., Chen, Q., Wu, G., Li, S., Gao, Z., Sun, L., Yu, B., Guo, R.: From words to structured visuals: A benchmark and framework for text-to-diagram generation and editing. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 13315–13325 (2025)
41. Wu, R., Su, W., Liao, J.: Chat2svg: Vector graphics generation with large language models and image diffusion models. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 23690–23700 (2025)
42. Wu, R., Su, W., Ma, K., Liao, J.: Iconshop: Text-guided vector icon synthesis with autoregressive transformers. *ACM Transactions on Graphics (TOG)* **42**(6), 1–14 (2023)
43. Xiang, Q., Qu, H., Rahmani, H., Liu, J.: Is monitoring enough? strategic agent selection for stealthy attack in multi-agent discussions. arXiv preprint arXiv:2603.21194 (2026)
44. Xiang, Q., Qu, H., Rahmani, H., Liu, J.: Ndpp-grasp: Non-differentiable physical plausibility constraint-guided task-oriented dexterous grasp generation. arXiv preprint arXiv:2606.02432 (2026)
45. Xing, X., Guan, Y., Zhang, J., Xu, D., Yu, Q.: Reason-svg: Hybrid reward rl for aha-moments in vector graphics generation. arXiv preprint arXiv:2505.24499 (2025)
46. Xing, X., Hu, J., Liang, G., Zhang, J., Xu, D., Yu, Q.: Empowering llms to understand and generate complex vector graphics. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 19487–19497 (2025)
47. Xue, Y., Qu, H., Li, Z., Lou, Y., Bai, Y., Rahmani, H., Liu, J.: Toolfg: Towards well-grounded fine-grained image classification. arXiv preprint arXiv:2606.02518 (2026)
48. Yang, Y., Cheng, W., Chen, S., Zeng, X., Yin, F., Zhang, J., Wang, L., YU, G., Ma, X., Jiang, Y.G.: Omnisvg: A unified scalable vector graphics generation model. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025)
49. Yin, S., Fu, C., Zhao, S., Li, K., Sun, X., Xu, T., Chen, E.: A survey on multimodal large language models. *National Science Review* **11**(12), nwae403 (2024)
50. Zala, A., Lin, H., Cho, J., Bansal, M.: Diagrammergpt: Generating open-domain, open-platform diagrams via llm planning. In: First Conference on Language Modeling (2023)
51. Zhang, H., Li, Z., Liu, J.: Scenellm: Implicit language reasoning in llm for dynamic scene graph generation. *Pattern Recognition* **170**, 111992 (2026)
52. Zhang, P., Zhao, N., Liao, J.: Text-to-vector generation with neural path representation. *ACM Transactions on Graphics (TOG)* **43**(4), 1–13 (2024)

53. Zhang, T., Wang, X., Li, L., Tai, Z., Chi, J., Tian, J., He, H., Wang, S.: STRICT: Stress-test of rendering image containing text. In: Christodoulopoulos, C., Chakraborty, T., Rose, C., Peng, V. (eds.) *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*. pp. 21137–21150. Association for Computational Linguistics, Suzhou, China (Nov 2025). <https://doi.org/10.18653/v1/2025.emnlp-main.1070>
54. Zhang, T., Lin, H., Liu, Z., Chen, C., Zhang, W.: Sciflow-bench: Evaluating structure-aware scientific diagram generation via inverse parsing. *arXiv preprint arXiv:2602.09809* (2026)
55. Zhang, Z., Wang, Y., Liu, J., Sun, G., Tang, K.: A two-phase kriging-assisted evolutionary algorithm for expensive constrained multiobjective optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* **54**(8), 4579–4591 (2024)
56. Zhang, Z., Wang, Y., Sun, G., Pang, T., Tang, K.: Constrained probabilistic pareto dominance for expensive constrained multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* (2024)
57. Zhu, D., Meng, R., Song, Y., Wei, X., Li, S., Pfister, T., Yoon, J.: Paperbanana: Automating academic illustration for ai scientists. *arXiv preprint arXiv:2601.23265* (2026)
58. Zhu, M., Lin, Z., Weng, Y., Lu, P., Xie, Q., Wei, Y., Liu, S., Sun, Q., Zhang, Y.: Autofigure: Generating and refining publication-ready scientific illustrations. In: *The Fourteenth International Conference on Learning Representations* (2026)
59. Zuo, J., Deng, H., Zhou, H., Zhu, J., Zhang, Y., Zhang, Y., Yan, Y., Huang, K., Chen, W., Deng, Y., et al.: Is nano banana pro a low-level vision all-rounder? a comprehensive evaluation on 14 tasks and 40 datasets. *arXiv preprint arXiv:2512.15110* (2025)