SKETCH-PLOT: Progressive Editing for Text-to-Image Academic Figures

Yinghao Tang
State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China
yinghaotang@zju.edu.cn

Yupeng Xie
HKUST(GZ)
Guangzhou, China
yxie740@connect.hkust-gz.edu.cn

Yingchaojie Feng
National University of Singapore
Singapore, Singapore
feng.y@nus.edu.sg

Tingfeng Lan
University of Virginia
Charlottesville, VA, USA
tingfeng@virginia.edu

Jiale Lao
Cornell University
Ithaca, NY, USA
jiale@cs.cornell.edu

Wei Chen
State Key Lab of CAD&CG, Zhejiang University
Hangzhou, China
chenvis@zju.edu.cn

Abstract

Text to image (T2I) models such as `gpt-image-2` can now generate publication grade academic figures from a short prompt, but the output is a flat raster: a user who wants to change one arrow, one label, or one icon has to regenerate the whole image, which also disturbs the parts they wanted to keep. We present SKETCH-PLOT, an interactive system that closes this controllability gap with a three layer progressive editing pipeline: a generated PNG, an addressable puzzle of editable pieces, and a per piece SVG. The user stops at the layer that gives them enough control for the change at hand, so the cost of decomposition and vectorisation is paid only on the pieces that need it. Realising this pipeline is not trivial. General segmentation models lack the semantic discriminability to decompose a research figure cleanly, and end to end image vectorisation produces incomplete shapes and loses semantic structure. We therefore route both stages through a **human in the loop** interface that lets the user accept, refine, or reject decomposition and vectorisation decisions on a piece by piece basis. We validate the design with an expert user study, in which participants found SKETCH-PLOT effective for making targeted edits to AI generated academic figures and preferred it over regenerating the whole image. A demonstration video is available at **https://paper-plot.dev/sketch**.

CCS Concepts

• **Human-centered computing** → **Human computer interaction (HCI)**; • **Computing methodologies** → **Artificial intelligence**.

KDD Workshop'26, Toronto, Canada

ACM ISBN 978-1-4503-XXXX-X/2026/08
https://doi.org/XXXXXXX.XXXXXXX

Keywords

academic figures, text-to-image, interactive editing, vision-language models, human-AI collaboration

ACM Reference Format:
Yinghao Tang, Yupeng Xie, Yingchaojie Feng, Tingfeng Lan, Jiale Lao, and Wei Chen. 2026. SKETCH-PLOT: Progressive Editing for Text-to-Image Academic Figures. In *Proceedings of KDD 2026 Workshop on AI Data Scientist (KDD Workshop'26).* ACM, New York, NY, USA, 6 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Academic figures such as flowcharts, pipeline diagrams, and architecture overviews are essential for communicating complex ideas in scholarly papers [5, 7, 10, 16]. However, creating these figures remains time consuming, often requiring hours of manual layout in tools such as PowerPoint, Keynote, or TikZ [2].

A promising approach is to delegate this work to text to image (T2I) models. Modern T2I models such as `gpt-image-2` [12], `qwen-image` [1], and `gemini-2.5-flash-image` [8] can produce visually polished figures from a short text prompt in seconds, and recent benchmarks confirm that the strongest among them reach publication grade quality on a substantial fraction of cases [11, 15]. As Figure 1 illustrates, this creates a tradeoff between slow but editable manual authoring and fast but hard to control generation. The fundamental limitation is that the generated output is a flat raster with no exposed structure. As a consequence, even a minor correction, such as fixing a label or changing an arrow direction, requires regenerating the entire figure, which also alters the parts that were already correct. The result is a polished figure that is not editable, a property we refer to as the *controllability gap* of T2I generation.

In this paper, we aim to solve this gap with a three layer progressive controllability pipeline, shown in the bottom row of Figure 1. The first layer is the raster PNG returned by the T2I model. The second layer is a *puzzle*: a complete partition of the canvas into addressable pieces that can be hovered, selected, resized along shared edges, copied, and re prompted, while the rest of the image stays fixed at the pixel level. The third layer is an SVG view in which a piece's text, colours, and strokes become attributes that the user can edit directly, without invoking any image model. The user stops

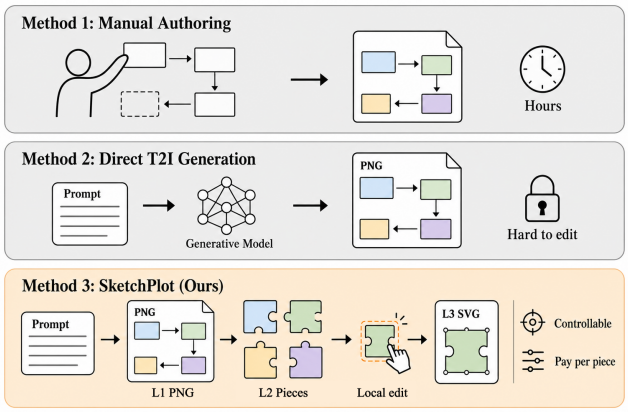

Figure 1: Three approaches to creating editable academic figures: manual authoring, direct text-to-image generation, and human-guided progressive editing through SKETCH-PLOT.

at the layer that gives them enough control for the change at hand, so the cost of decomposition and vectorisation is paid only on the pieces that need it.

Realising this pipeline is not trivial. First, general segmentation backbones lack the semantic discriminability to recover panels, header bars, and labelled boxes from a research figure as coherent units [6]. Second, end to end image vectorisation, applied to a whole figure, produces incomplete shapes and loses semantic structure [3]. We address both with a **human in the loop** interface that sits between the user and the underlying models. For the first challenge, instead of asking a segmentation model to produce a perfect decomposition, we ask a vision language model for a coarse layout and let the user refine it directly in the puzzle interface, fixing any decomposition mistake before moving on. For the second, instead of vectorising the whole figure in one shot, we expose vectorisation as a per piece operation invoked only on the pieces the user selects. Together, these mechanisms give the user fine grained control over what gets edited and at what fidelity, while keeping any model error local to a single piece rather than the whole figure.

We summarise our contributions as follows:

- We propose a three layer progressive controllability pipeline (PNG, puzzle, SVG) for post hoc editing of T2I generated academic figures, with a human in the loop design that sidesteps the failure modes of end to end segmentation and vectorisation.
- We present SKETCH-PLOT, an open source web based instantiation of the pipeline that integrates a T2I model, a vision language model for layout, and a per piece vectoriser into a single editing interface.
- We validate the design with an expert user study, in which participants found SKETCH-PLOT effective for making targeted edits to AI generated academic figures and preferred it over regenerating the whole image.

A demonstration video is available at **https://paper-plot.dev/sketch**.

2 Related Work

Text-to-Image Figure Generation. Recent text-to-image (T2I) models [8, 12, 14], such as gpt-image-2 [12] and gemini-2.5-flash-image [8], can produce visually polished academic figures from a short prompt in seconds, and dedicated reliability benchmarks have begun to assess them, e.g., IGenBench [15]. Their output, however, is a flat raster with no exposed structure: individual elements cannot be addressed, and correcting even a single label requires regenerating the whole figure, which alters parts the author wished to preserve. Code-driven alternatives [2], such as AutomaTikZ [2], restore full editability by generating TikZ source, but the visual style is bound by what the underlying language can express and the authoring curve remains steep. SKETCH-PLOT sits at the opposite end of this trade-off: visual quality comes from the T2I model, and structure is recovered afterwards.

Image Editing and Segmentation. Mask-conditioned image editing methods [4, 12], such as InstructPix2Pix [4], allow a user to modify a region of an image given a mask and a text instruction, but the mask itself must be painted by hand without any knowledge of the figure's semantic structure. General-purpose segmentation models [9, 13], such as SAM [9], produce accurate pixel-level masks yet over-segment text into individual glyphs, miss thin arrows below their area threshold, and emit no semantic label for what a region means. SKETCH-PLOT supplies the missing layer: a VLM-derived, semantically named partition of the canvas that lets the user address a region by its semantic role rather than by a brush stroke. This named partition directly enables region-level inpainting: the user selects a region by name and issues an edit prompt, and its bounding box automatically serves as the mask.

3 System

3.1 Overview

SKETCH-PLOT is organised as a three-layer pipeline driven by a single human-in-the-loop interface (Figure 2). Given a short text prompt, layer **L1** calls a text-to-image model and returns a flat raster figure. Our current implementation routes through OpenRouter and uses `gpt-image-2` at 1536×1024, but the backend is interchangeable and we have also `gemini-3.1-flash-image-preview`. The raster looks finished but is opaque. Every box, arrow, and label is fused into a single PNG, so any subsequent edit forces the user back to the prompt. Layer **L2** closes that gap. A vision-language model (default `gemini-3.5-flash`) is asked to read the raster and emit a coarse layout: a small set of three to ten top-level bounding boxes with semantic labels (e.g. `header`, `cards_row`, `cost_panel`). Layer **L3** is invoked on demand. For any selected piece, the system runs an image-to-SVG pass that converts the local raster region into editable vector primitives, so text, stroke, fill, and geometry become first-class handles instead of pixels.

The VLM output by itself is not yet a controllable surface. The returned boxes have gaps, occasional overlaps, and ignore everything the model judged unimportant, so they cannot be used as a partition. We close the layout into a *puzzle* with a deterministic residual-completion step. Named regions are sorted by area in ascending order so that smaller, more specific boxes win any overlap with their larger ancestors. We then walk the regions in that order

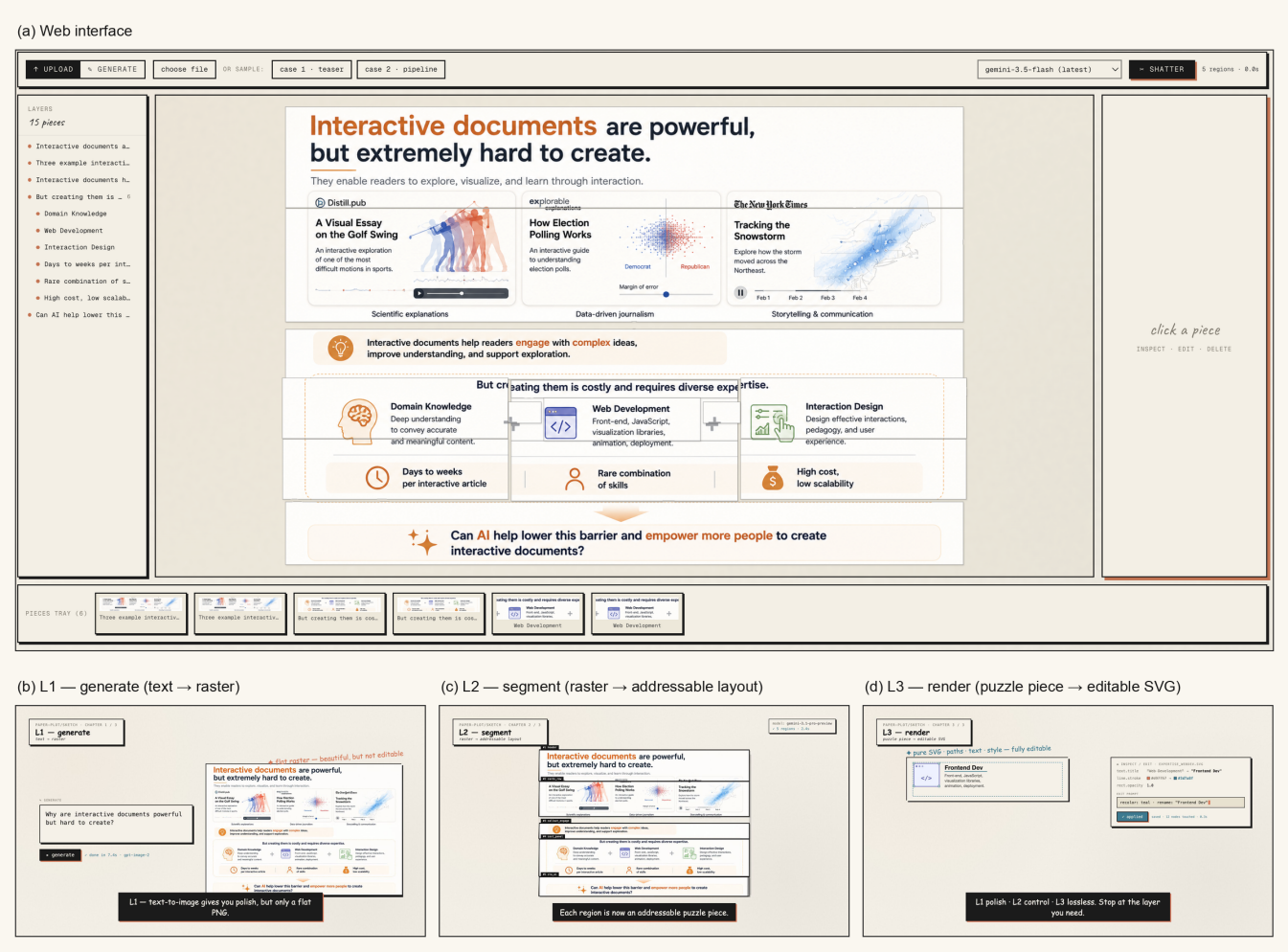

Figure 2: System overview of SKETCH-PLOT. (a) The web interface exposes the three layers through a single canvas: a top bar drives generation and the SHATTER action; the centre canvas hosts the raster figure; the left *layers* panel and bottom *pieces tray* expose the segmented hierarchy; the right inspector hosts per-piece *inspect* / *edit* / *delete* actions. (b, c, d) The three progressive editing stages. L1 (generate) turns a text prompt into a flat raster figure. L2 (segment) asks a VLM to propose regions and uses a residual-completion pass to turn the raster into a canvas of addressable *puzzle pieces*. L3 (render) vectorises a selected piece on demand, exposing text, stroke, and style as first-class handles. Users stop at whichever layer their task demands.

and clip each one to the still-free area of the canvas using axis-aligned rectangle subtraction, which keeps the largest contiguous fragment as the visible piece for that label and re-emits any leftover slivers as background pieces. Whatever free area remains at the end is sliced into background rectangles as well. The output is an exact tiling of the canvas: every pixel belongs to exactly one piece, every piece carries either a semantic label or a `background` tag, and every piece exposes the same interaction vocabulary. This invariant is what lets the rest of the system treat "imperfect VLM output" as a feature rather than a blocker, since the user can absorb, split, or re-label background pieces without ever leaving the canvas.

3.2 User Interface

The interface (Figure 2a) is a single-page canvas organised into four regions. The **top bar** hosts the three pipeline controls: UPLOAD an existing figure or GENERATE one from a prompt, pick a backbone model, and trigger SHATTER (the L2 segmentation pass). The **centre canvas** always shows the current raster, with each puzzle piece drawn as a translucent overlay so that hovering reveals the piece boundary and clicking selects it. The **layers panel** on the left and the **pieces tray** at the bottom both expose the same L2 hierarchy: the panel as a textual outline (for navigation by label), the tray as thumbnails (for navigation by appearance). The **inspector** on the right is the editing surface for the currently selected piece.

The inspector exposes four actions on a piece, ordered by increasing commitment. *Inspect* reveals the piece's metadata (label,

type, bounding box, parent) and a thumbnail crop. *Drag-edge* lets the user nudge the boundary shared with a neighbour. The drag is resolved by re-running TILIFY with the dragged region pinned to its new bounds, so neighbours give way and the canvas is re-tiled in a single pass without leaving holes. *Re-prompt* sends the piece's crop, mask, and a short instruction to an image-edit model and pastes the result back into that region only, leaving every other piece untouched at the pixel level. *Vectorise* invokes L3 and replaces the piece's raster with editable SVG. Because all four regions stay live at once, the user can move between layers (regenerate L1 from a revised prompt, re-shatter L2 with a different backbone, vectorise an L3 piece) without leaving the page or losing the rest of the canvas.

4 User Study

We conducted a user study to evaluate whether SKETCH-PLOT helps users create and refine AI-generated academic figures in a controllable way across its L1–L3 workflow.

Participants. We recruited three participants from artificial intelligence and human-computer interaction. All participants had created figures for academic papers or presentations, and none had used SKETCH-PLOT before the study.

Procedure. Each session lasted approximately 40 minutes. We first gave participants a brief introduction to SKETCH-PLOT and demonstrated the basic interaction flow. Participants then completed the figure authoring task while thinking aloud. After the task, they answered five questions on a 5-point Likert scale, covering ease of learning, ease of use, the L1–L3 workflow, the L2 decomposition stage, and the L3 rendering stage, and joined a short semi-structured interview about their experience.

Tasks. Participants were asked to create an academic figure based on their own research work using SKETCH-PLOT. The target figure could be a method overview, pipeline diagram, system architecture figure, or conceptual framework. Participants first described the figure they wanted to create, then used SKETCH-PLOT to generate an initial figure and refine selected parts through the L1–L3 workflow.

Results and Feedback. Overall, participants responded positively to SKETCH-PLOT. As shown in Figure 3, participants rated the system highly on ease of learning and ease of use, with both questions receiving a mean score of 5.0. They also agreed that the L1–L3 workflow helped them turn a research idea into an editable academic figure (4.67 ± 0.58), and that the L2 decomposition stage helped preserve and reuse satisfactory parts of the figure (4.67 ± 0.58). Participants gave a slightly lower but still positive rating to the L3 rendering stage for converting selected raster pieces into SVG elements for further editing and reuse (4.33 ± 0.58). In the interviews, participants appreciated that useful parts of a generated figure could be kept as reusable pieces instead of being lost during regeneration. They also found the decomposition helpful for reasoning about the figure as a set of academic visual units, such as panels, labels, icons, and arrows. For L3, participants valued the possibility of moving selected parts into an editable SVG form, while noting that complex icons or dense visual regions sometimes still required additional refinement.

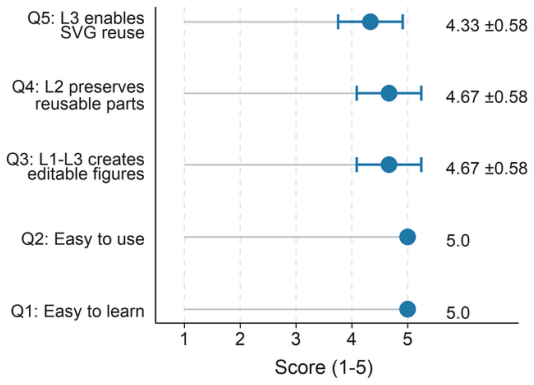

Figure 3: Participant ratings of SKETCH-PLOT on five study questions. Values show mean and standard deviation.

User Suggestions. Participants suggested several improvements. They wanted an option to further decompose a selected region for finer control inside a larger panel, more precise boundary snapping for dense, text-heavy layouts, and editing affordances such as undo, edit preview, and version history.

5 Future Work

We see SKETCH-PLOT as a step toward a broader practice for authoring academic figures in which T2I and VLM models act as collaborators the author can steer at the level of individual pieces. We hope the puzzle paradigm extends beyond the flowcharts and architecture diagrams that drove the current design. Tables, experimental plots, and code-driven figures share the same controllability gap, but each calls for a different vocabulary of editable units: rows, headers, and cell groups for tables; axes, series, and annotations for plots; and the underlying script alongside the rendered image for code-driven figures. Working out these per-domain vocabularies and their associated best practices is the direction we plan to pursue next.

Two known limitations point to concrete next steps. First, current VLMs are still imperfect at parsing structured visual content [6]: panels can be fused, sub-panels dropped, and bounding boxes are sometimes a few pixels short. The puzzle absorbs these failures as background pieces the user can fix, and we plan to reduce the cost with prompting refinements and a pass that snaps box edges to image gradients. Second, image-to-SVG conversion in general remains weak on fine edge detail such as icons, glyphs, and thin connectors [2, 3]; closing this gap as the underlying models improve is a direction we plan to track.

6 Conclusion

We proposed the puzzle paradigm for post hoc editing of T2I generated academic figures, defined by two principles: every pixel belongs to exactly one editable piece, and every piece exposes the same interaction vocabulary. The paradigm reframes layout extraction as "any layout, plus a residual completion" rather than "perfect layout", so imperfect VLM output stops being a blocker

for a controllable editing surface. We instantiated the paradigm in SKETCH-PLOT, a three layer progressive editing pipeline (PNG, puzzle, SVG) wrapped in a human in the loop interface, and validated the design with an expert user study in which participants found SKETCH-PLOT effective for making targeted edits to AI generated academic figures and preferred it over regenerating the whole image.

References

[1] Alibaba DAMO. 2025. Qwen-Image and Qwen-Image-Edit. https://qwenlm.github.io/.

[2] Jonas Belouadi, Anne Lauscher, and Steffen Eger. 2024. AutomaTikZ: Text-Guided Synthesis of Scientific Vector Graphics with TikZ. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[3] Qi Bing, Chaoyi Zhang, and Weidong Cai. 2024. DeepIcon: A Hierarchical Network for Layer-wise Icon Vectorization. In *International Conference on Digital Image Computing: Techniques and Applications (DICTA)*.

[4] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[5] Yiyu Chen, Yifan Wu, Shuyu Shen, Yupeng Xie, Leixian Shen, Hui Xiong, and Yuyu Luo. 2025. ChartMark: A Structured Grammar for Chart Annotation. In *2025 IEEE Visualization and Visual Analytics (VIS)*. IEEE, 311–315.

[6] Miguel Espinosa, Chenhongyi Yang, Linus Ericsson, Steven McDonagh, and Elliot J. Crowley. 2024. There is no SAMantics! Exploring SAM as a Backbone for Visual Understanding Tasks. *arXiv preprint arXiv:2411.15288* (2024).

[7] Steven L. Franconeri, Lace M. Padilla, Priti Shah, Jeffrey M. Zacks, and Jessica Hullman. 2021. The Science of Visual Data Communication: What Works. *Psychological Science in the Public Interest* 22, 3 (2021), 110–161.

[8] Google DeepMind. 2026. Gemini image generation: `gemini-2.5-flash-image` and `gemini-3.1-flash-image-preview`. https://ai.google.dev/.

[9] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. 2023. Segment Anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[10] Bongshin Lee, Nathalie Henry Riche, Petra Isenberg, and Sheelagh Carpendale. 2015. More Than Telling a Story: Transforming Data into Visually Shared Stories. *IEEE Computer Graphics and Applications* 35, 5 (2015), 84–90.

[11] Boyan Li, Yiran Peng, Yupeng Xie, Sirong Lu, Yizhang Zhu, Xing Mu, Xinyu Liu, and Yuyu Luo. 2026. Deepeye: A steerable self-driving data agent system. *arXiv preprint arXiv:2603.28889* (2026).

[12] OpenAI. 2026. GPT Image 2: Image generation and editing API. https://platform.openai.com/docs/models/gpt-image-2.

[13] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. 2024. SAM 2: Segment Anything in Images and Videos. https://arxiv.org/abs/2408.00714.

[14] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10684–10695.

[15] Yinghao Tang, Xueding Liu, Boyuan Zhang, Tingfeng Lan, Yupeng Xie, Jiale Lao, Yiyao Wang, Haoxuan Li, Tingting Gao, Bo Pan, Luoxuan Weng, Xiuqi Huang, Minfeng Zhu, Yingchaojie Feng, Yuyu Luo, and Wei Chen. 2026. IGenBench: Benchmarking the Reliability of Text-to-Infographic Generation. *arXiv preprint arXiv:2601.04498* (2026).

[16] Yinghao Tang, Yupeng Xie, Yingchaojie Feng, Tingfeng Lan, Jiale Lao, Yue Cheng, and Wei Chen. 2026. ViviDoc: Generating Interactive Documents through Human-Agent Collaboration. *arXiv preprint arXiv:2603.27991* (2026).