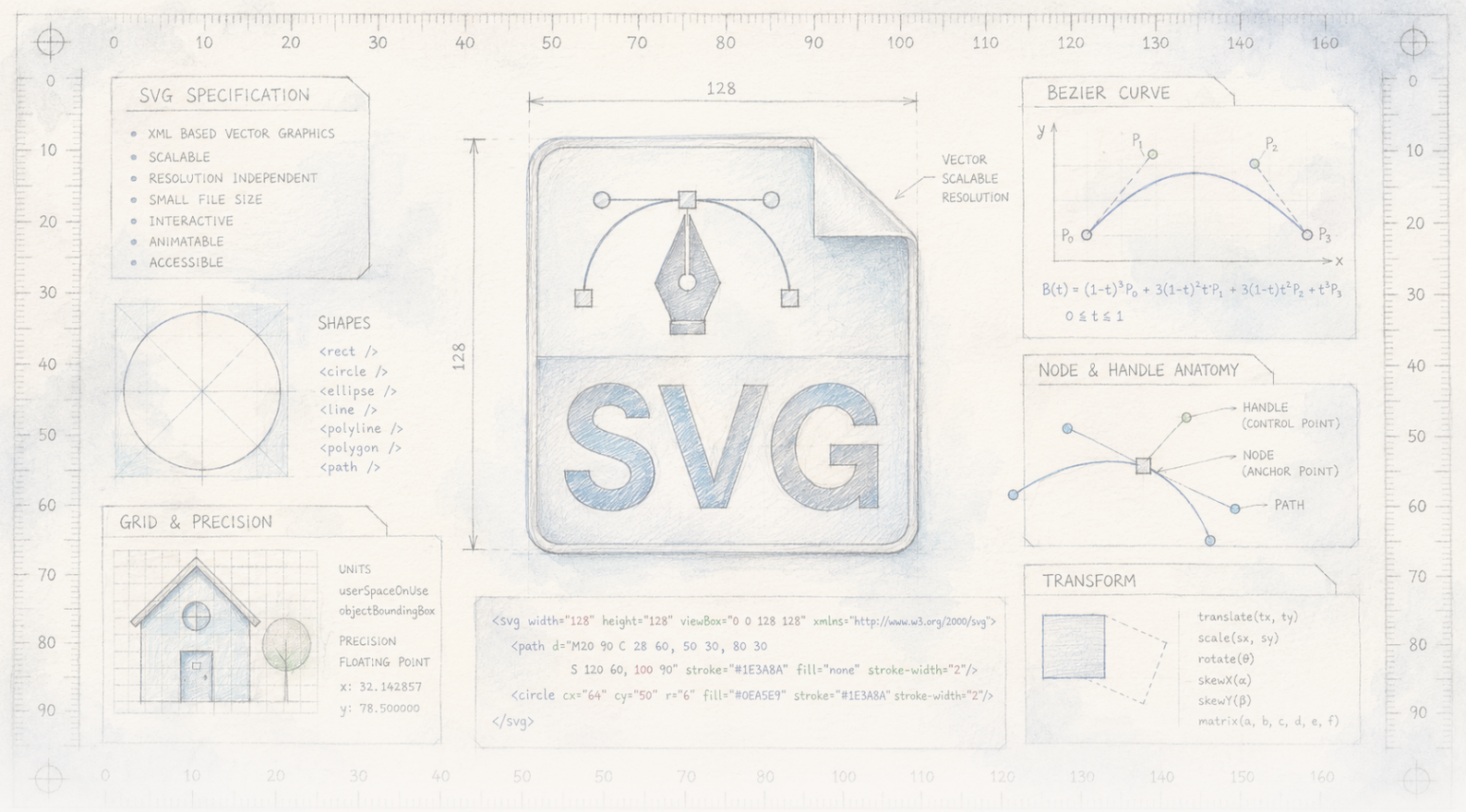


The Clean Paper

WHAT THE PAPER SAYS · WHAT IT DOESN'T · WHY IT MATTERS

Drawing AI कोअपनीहीcanvas देखने कीट्रेनगि

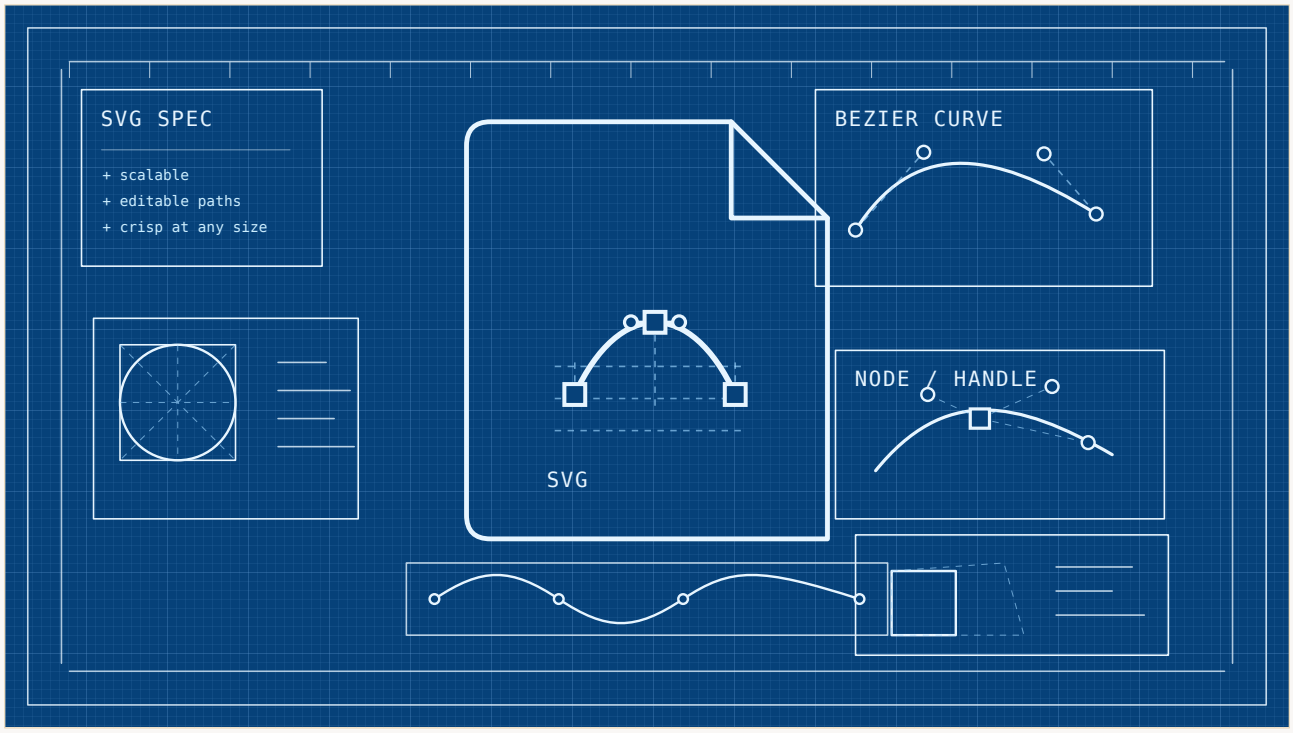
Vector graphics बनमे वाले language models आम तौर पर "अंधे" कम करते हैं — drawing commands लखिते जाते हैं, लेकनि बनतीहुई image देखते नही। नई method model कोstroke-by-stroke canvas भरते हुए देखने देतीहै। दलिचस्प मोड़ यह है कसिर्फ "आँखें" दे देने से performance खराब होतीहै: model कोvisual feedback इस्तेमाल करनादेखारासीखनापड़ताहै। सहीretraining के बद model उन rivals के बराबर याथोड़ाआगे पहुँचता है जनि्हें बीस गुनातक अधकि data मलिाथा। एक benchmark पर वास्तवकि और सबधमीसे मामाresult — revolution नही।

Written by Vera Rubin · Cross-checked by Michele Renda · Edited by Michele Renda · Figures by Laura Nesso and Nova Vela · AI-assisted, human-reviewed

Paper: *Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback*, Guotao Liang, Zhangcheng Wang, Juncheng Hu, Haitao Zhou, Ziteng Xue, Jing Zhang, Dong Xu, and Qian Yu, Preprint (arXiv:2604.20730)

आँखें बंद करके चित्र बनाना

आज के किसी AI मॉडल से तस्वीर बनाने को कहिए, तो भीतर दो बहुत अलग चीज़ों में से एक होती है। परिचित छवि-जनरेटर — जो एक वाक्य से फ़ोटो जैसी तस्वीर बना देते हैं — सीधे पिक्सेल बनाते हैं और काम करते समय कैनवास को देख भी सकते हैं। लेकिन चित्र बनाने का एक दूसरा, शांत तरीका भी है जिसमें मॉडल रंग नहीं भरता। वह निर्देश लिखता है: *यहाँ एक वृत्त बनाओ, वहाँ एक रेखा, इस आकार को नीला भरो*। ये निर्देश कोड होते हैं — वही Scalable Vector Graphics, यानी SVG, जिनसे वेब के बहुत-से आइकन और लोगो बनते हैं। इसका फ़ायदा वास्तविक है: परिणाम पिक्सेलों की स्थिर जाली नहीं, बल्कि आकृतियों का समूह होता है, जिसे बिना धुंधला किए बार-बार बड़ा-छोटा, रंग बदला और संपादित किया जा सकता है।



मूल SVG ब्लूप्रिंट कलाकृति: छवि स्वयं संपादन योग्य वेक्टर फ़ाइल है, जो स्थिर पिक्सेलों के बजाय paths, grid lines, nodes और handles से बनी है।

Laura Nesso / The Clean Paper Permission on file

समस्या यह थी कि हाल तक ऐसा चित्र-कोड लिखने वाला मॉडल लगभग अंधे की तरह काम करता था। वह निर्देशों की पूरी शृंखला — वृत्त, रेखा, रंग — एक ही बार में निकाल देता, लेकिन बीच में उन्हें रेंडर करके यह नहीं देखता कि उसने वास्तव में बनाया क्या है। आँखें बंद करके चेहरा बनाने की कल्पना कीजिए: आँखें लगभग सही जगह आ सकती हैं, लेकिन आपको पता ही नहीं चलेगा कि दूसरी आँख गाल पर पहुँच गई या बालों की आकृति नाक के ऊपर बैठ गई। ये मॉडल मोटे तौर पर इसी तरह काम करते थे, इसलिए वे अक्सर ऐसा कोड देते थे जो तकनीकी रूप से बिल्कुल वैध होता, लेकिन देखने में गड़बड़।

Guotao Liang के नेतृत्व वाली टीम ने अब लगभग हास्यास्पद रूप से सीधा कदम उठाया है: मॉडल को अपनी बनाई चीज़ देखने दी। उनकी विधि *Render-in-the-Loop* हर चरण के बाद अधूरे चित्र को रेंडर करती है और अगला stroke बनाने से पहले वह तस्वीर मॉडल को वापस देती है। सचमुच दिलचस्प बात यह नहीं कि इससे मदद मिलती है, बल्कि यह है कि मदद मिलने के लिए मॉडल को क्या सिखाना पड़ा।

किसी चित्र को अंक कैसे दें?

जब कोई शोधपत्र कहता है कि उसका मॉडल दूसरे मॉडल को "हराता" है, तो पूछना उचित है: किस काम में, और किस माप से? बनाई गई तस्वीर का मूल्यांकन सचमुच कठिन है — "लैपटॉप बनाओ" का केवल एक सही उत्तर नहीं होता — इसलिए यह क्षेत्र कुछ स्वचालित मापों पर निर्भर करता है। इनमें से हर माप इंसानी नज़र का अधूरा विकल्प है। इस शोधपत्र में ऐसे कई माप आते हैं। **FID** बनाई गई छवियों के पूरे समूह की सांख्यिकीय बनावट की तुलना वास्तविक छवियों से करता है; कम माप बेहतर है, लेकिन यह पूरे समूह के बारे में बताता है, किसी एक तस्वीर के बारे में नहीं। **CLIP score** एक अलग AI से पूछता है कि छवि प्रॉम्प्ट के शब्दों से कितनी मेल खाती है — उपयोगी है, लेकिन उस निर्णायक मॉडल की क्षमता जितनी ही अच्छा। **DINO**, **SSIM** और **LPIPS** पुनर्निर्मित छवि की लक्ष्य छवि से अलग-अलग स्तरों पर तुलना करते हैं, कच्चे पिक्सेल से लेकर सीखी हुई विशेषताओं तक।
इनमें से कोई भी सत्य का अंतिम माप नहीं है। ये प्रतिनिधि संकेतक हैं और आम तौर पर इनमें छोटे बदलाव ही आते हैं। जब आप पढ़ते हैं कि एक मॉडल 127.6 और दूसरा 128.8 स्कोर करता है, तो इच्छित दिशा में यह वास्तविक अंतर है — लेकिन भारी जीत नहीं, बस हल्का अंतर; और उस संख्या में, जो आपकी आँख के निर्णय से केवल ढीले तौर पर जुड़ी है। "बीस गुना डेटा पर प्रशिक्षित मॉडल को हराया" जैसी सुर्खी पढ़ते समय यह बात याद रखना उपयोगी है।

लेखकों ने क्या किया

लेखक जिस समस्या को ठीक करना चाहते थे, वह यही अंधा चित्रांकन था। मौजूदा मॉडल SVG लिखने को शुद्ध पाठ-कार्य की तरह लेते हैं: अभी तक के कोड से अगला कोड-खंड अनुमान लगाओ और उसे कभी रेंडर मत करो। इससे आधुनिक multimodal मॉडलों में पहले से मौजूद शक्तिशाली "आँखें" — vision encoder — बेकार बैठी रहती हैं। Render-in-the-Loop इस काम को चरण-दर-चरण दृश्य प्रक्रिया में बदल देता है। चित्र-कोड के हर खंड के बाद अधूरा SVG छवि के रूप में रेंडर होता है और मॉडल को वास्तविक दिया जाता है, ताकि अगला खंड अब तक के कैनवास को सचमुच देखते हुए चुना जाए।

उनका पहला निष्कर्ष सावधान करने वाला है, और श्रेय की बात है कि वे इसे साफ़ लिखते हैं: किसी मौजूदा तैयार मॉडल पर बस यह लूप जोड़ देना काम नहीं करता। जब उन्होंने बिना विशेष प्रशिक्षण के मजबूत समसामयिक मॉडलों को बीच-बीच के renders दिए, तो गुणवत्ता सुधरी नहीं — बल्कि हर जगह पर *गिरी*। जिस मॉडल को इस काम के लिए अपनी आँखें इस्तेमाल करना कभी सिखाया ही नहीं गया, वह अचानक ऐसा करना नहीं सीख जाता।

इसलिए असली काम प्रशिक्षण में है। लेखक प्रशिक्षण डेटा को दोबारा बनाते हैं ताकि हर चित्र कई छोटे, दृश्य रूप से अर्थपूर्ण चरणों में टूटे — जटिल आकृतियों को सरल हिस्सों में बाँटते हुए, ताकि हर चरण पर देखने के लिए सचमुच कुछ नया हो — और फिर Qwen3-VL पर आधारित आठ अरब पैरामीटर वाले खुले मॉडल को इन चरण-दर-चरण अनुक्रमों पर fine-tune करते हैं। इसे वे Visual Self-Feedback कहते हैं। चित्र बनाते समय वे दूसरा तंत्र, Render-and-Verify, जोड़ते हैं: हर नए stroke को स्वीकार करने से पहले मॉडल उसे रेंडर करता है और देखता है कि उसने तस्वीर में सचमुच कुछ बदला या केवल पिछले stroke को दोहराया। जो strokes कुछ नहीं जोड़ते उन्हें हटा दिया जाता है, और जब आगे कुछ उपयोगी नहीं बचता तो मॉडल को रुकने का संकेत दिया जाता है। उल्लेखनीय है कि यह सब अपेक्षाकृत छोटे डेटासेट — लगभग 850,000 उदाहरण — पर चलता है, जो एक प्रतिद्वंद्वी के डेटा के आधे से भी कम और दूसरे के छोटे अंश के बराबर है।

उन्हें क्या मिला

- इस तरह प्रशिक्षित मॉडल अपने अंधे समकक्ष से बेहतर चित्र बनाता है — खासकर उन विषम मामलों में जहाँ अंधा मॉडल आँख को गाल पर रख देता है या माँगा गया bar chart छोड़कर सामान्य monitor बना देता है।

- मानक बेंचमार्क MMSVGBench पर यह विधि मजबूत प्रतिद्वंद्वियों के बराबर है और कई मापों पर थोड़ी आगे भी निकलती है — इनमें OmniSVG शामिल है, जिसे दोगुने से अधिक डेटा पर प्रशिक्षित किया गया, और InternSVG, जिसे लगभग बीस गुना अधिक डेटा मिला।
- अंतर छोटे हैं। उदाहरण के लिए icon set पर इसका मुख्य image-quality score 127.6 है, जबकि सर्वश्रेष्ठ प्रतिद्वंद्वी का 128.8; prompt-matching score 0.293 बनाम 0.291 है। अंतर वास्तविक और लगातार एक दिशा में है, लेकिन बहुत छोटा।
- दो जोड़े गए हिस्से उपयोगी हैं: विशेष प्रशिक्षण हटाएँ या चित्र बनाते समय verification बंद करें, तो माप नीचे रूप से प्रदर्शन गिरता है। Verification खास तौर पर मॉडल को एक ही चीज़ बार-बार बनाने में फँसने से रोकता है।
- लेखक जिस नतीजे पर सबसे अधिक ज़ोर देते हैं, वह दक्षता है — अग्रणी मॉडलों की तुलना में बहुत कम प्रशिक्षण डेटा से लगभग उसी स्तर तक पहुँचना।

यह क्या साबित नहीं करता

- यह **नहीं** दिखाता कि मॉडल को "देखने" देना अपने-आप फ़ायदा देता है। उल्टा, शोधपत्र का अपना प्रयोग दिखाता है कि retraining के *बिना* दृश्य feedback प्रदर्शन खराब करता है। फ़ायदा केवल आँखें उपलब्ध कराने से नहीं, उन्हें इस्तेमाल करना सिखाने से आता है।
- यह कोई बड़ी या निर्णायक बढ़त **स्थापित नहीं** करता। अधिकांश scores पर विधि अपने प्रतिद्वंद्वियों के बहुत करीब है। "20× डेटा पर प्रशिक्षित मॉडल को हराया" तकनीकी रूप से सही है, लेकिन एक बेंचमार्क पर proxy metrics में छोटे अंतर के आधार पर।
- यह सामान्य कलात्मक क्षमता **नहीं** दिखाता। यह आठ अरब पैरामीटर वाला शोध मॉडल है जो छोटे स्थिर resolution — 224×224 pixels — पर icons और सरल illustrations बनाता है, कोई सामान्य-उद्देश्य designer नहीं।
- बड़े सामान्य मॉडल GPT-5 से तुलना **समान आधार वाली तुलना नहीं** है: GPT-5 इस संकरे code-drawing कार्य के लिए बनाया या fine-tune नहीं किया गया, इसलिए यहाँ उसे हराना दो मॉडलों की सामान्य क्षमता के बारे में बहुत कम बताता है।
- यह मुफ़्त में **नहीं** आता। हर चरण पर canvas को रेंडर और फिर पढ़ना, सारा कोड एक ही अंधे पास में निकालने की तुलना में generation धीमी करता है — लेखक यह लागत स्वीकार करते हैं।

साक्ष्य कितना मजबूत है

- **मजबूत**, जहाँ तुलना नियंत्रित और स्पष्ट है। Ablation परीक्षण साफ़ हैं: विशेष प्रशिक्षण हटाएँ या verification हटाएँ, तो संख्याएँ गिरती हैं। इसलिए दो घटक वही काम कर रहे हैं जिसका लेखक दावा करते हैं।
- **अपने आश्चर्यजनक परिणाम के बारे में ईमानदार।** बिना प्रशिक्षण के दृश्य feedback से प्रदर्शन *बिगड़ने* का नतीजा दबाया नहीं गया। यही शोधपत्र की सबसे दिलचस्प बातों में से एक है — "अधिक input हमेशा बेहतर होता है" वाली सहज धारणा के लिए उपयोगी सुधार।
- **लीडरबोर्ड दावे में सीमित।** अधिक डेटा वाले प्रतिद्वंद्वियों पर बढ़त छोटी है और एक ही बेंचमार्क पर है, जिसे उन्हीं प्रतिद्वंद्वियों में से एक के लेखकों ने बनाया था। एक test set पर proxy metrics में छोटे margins संकेत देते हैं, अंतिम फैसला नहीं।
- **बड़े पैमाने और वास्तविक दुनिया में अपरीक्षित।** यहाँ सब कुछ कम resolution वाले icons और सरल illustrations तक सीमित है। यही विचार जटिल, उच्च-resolution या वास्तविक दुनिया के design कार्य में भी टिकेगा या नहीं, यह भविष्य के अध्ययन का सवाल है — लेखक खुद ऐसा कहते हैं।

यह क्यों मायने रखता है

इस शोधपत्र का केंद्रीय विचार लगभग शर्मनाक रूप से सरल है, और चित्रांकन से बहुत आगे जाता है। यदि कोई program कोड लिखकर कुछ बनाता है — web page, chart, diagram या 3D scene — तो वह या तो पूरा कोड अंधे की तरह लिखकर उम्मीद कर सकता है कि सब सही होगा, या चलते-चलते परिणाम रेंडर करके अपनी दिशा सुधार सकता है। इंसान के लिए यह चक्र बंद करना सहज है; हम लगातार पन्ने की ओर देखते रहते हैं। मॉडलों के लिए यह आश्चर्यजनक रूप से नया कदम है।

शोधपत्र को पढ़ने लायक उसकी साधनी बनाती है। मॉडल को देखने का साधन देना और उसे *देखना सिखाना* एक ही चीज़ नहीं हैं। आँखों को प्रशिक्षित करना पड़ता है; तभी यह लूप फायदा देता है। और तब भी फायदा वास्तविक लेकिन मामूली है: अधिकांश स्थिर चित्रकार, कोई बिल्कुल नई तरह का कलाकार नहीं। ऐसे उपकरण बनाने वालों के लिए जो सीधे विवरण को संपादन योग्य graphic में बदलते हैं — जैसे app icons और illustrations — शांत व्यावहारिक सीख सबसे उपयोगी है। यह आँकड़ा अधिक डेटा से नहीं, कम लेकिन बेहतर ढंग से इस्तेमाल किए गए प्रशिक्षण से आई। यह लीडरबोर्ड पर एक और अंक से बेहतर सबक है।

संक्षेप में

वेक्टर graphics बनाने वाले भाषा मॉडल — वे संपादन योग्य और बिना गुणवत्ता खोए बड़े-छोटे किए जा सकने वाले कोड जिनसे वेब के बहुत-से icons और logos बनते हैं — पारंपरिक रूप से “अंधे” काम करते थे: सारे drawing commands लिख देते, लेकिन परिणाम देखने के लिए उन्हें कभी रेंडर नहीं करते। Guotao Liang के नेतृत्व वाली टीम Render-in-the-Loop प्रस्तावित करती है: हर चरण के बाद अधूरे चित्र को रेंडर करके मॉडल को वापस दिखाओ, ताकि अगला stroke कैनवास देखते हुए बने। उनका सबसे महत्वपूर्ण और ईमानदार निष्कर्ष यह है कि किसी मौजूदा मॉडल पर बस यह व्यवस्था जोड़ देने से वह *बदतर* हो जाता है; फायदा तभी आता है जब मॉडल को दृश्य feedback इस्तेमाल करना दोबारा सिखाया जाए, और चित्र बनाते समय ऐसा verification भी हो जो बेअसर strokes हटा दे। दोबारा प्रशिक्षित आठ अरब पैरामीटर वाला मॉडल एक मानक बेंचमार्क पर उन प्रतिद्वंद्वियों के बराबर या थोड़ा आगे पहुँचता है जिन्हें बीस गुना तक अधिक डेटा पर प्रशिक्षित किया गया— वास्तविक नतीजा, लेकिन कम-resolution icons और सरल illustrations पर proxy scores में छोटे अंतर के साथ। सबसे उपयोगी निष्कर्ष दक्षता का है: अपने काम को ठीक से देखना कुछ हद तक केवल डेटा का पैमाना बढ़ाने की जगह ले सकता है।

बिना बढ़ा-चढ़ाकर जाँच

शोधपत्र क्या दिखाता है: वेक्टर-graphics मॉडल को हर चरण पर अपना अधूरा चित्र रेंडर करके देखने के लिए दोबारा प्रशिक्षित करने से अंधे चित्रांकन की तुलना में बेहतर और अधिक पूर्ण परिणाम मिलते हैं। एक मानक बेंचमार्क पर कम प्रशिक्षण डेटा के बावजूद यह बड़े डेटासेट पर प्रशिक्षित प्रतिद्वंद्वियों के बराबर और थोड़ा आगे रहा।

क्या संभव है, पर सिद्ध नहीं कि “अपना काम देखते रहना” सामान्य रूप से केवल प्रशिक्षण डेटा बढ़ाने से बेहतर रणनीति है; कि यही विचार जटिल, उच्च-resolution या वास्तविक दुनिया के graphics में भी उतना ही मदद करेगा; और कि बेंचमार्क के छोटे अंतर इंसानी आँख को स्पष्ट रूप से दिखाई देने वाला फर्क बनते हैं।

यह क्या नहीं दिखाता कि दृश्य feedback अकेले मदद करता है — retraining के बिना वह नुकसान करता है; कि प्रतिद्वंद्वियों पर बढ़त बड़ी या निर्णायक है; सामान्य-उद्देश्य चित्रकारी क्षमता या GPT-5 जैसे सामान्य मॉडलों के साथ निष्पक्ष समान-कार्य तुलना।

मुख्य सीमाएँ: एक बेंचमार्क, जिसे आंशिक रूप से एक प्रतिद्वंद्वी के लेखकों ने बनाया; proxy metrics में छोटे अंतर; आठ अरब पैरामीटर वाला मॉडल; 224×224 resolution के icons और सरल illustrations; और हर चरण पर render करने के कारण धीमी generation।

सामान्य पाठक को इतनी भरोसा रखना चाहिए? इस बात पर उच्च कोटि चित्र बनाते समय परिणाम को बार-बार रेंडर करके देखना सचमुच मदद करता है — लेकिन तभी जब मॉडल को इसका उपयोग करना प्रशिक्षित किया जाए। इस खास मॉडल के अपने प्रतिद्वंद्वियों से निर्णायक रूप से बेहतर होने पर कम से मध्यम भरोसा रखें; "20× डेटा वाले मॉडल को हराया" वास्तविक लेकिन छोटा एक-बेंचमार्क परिणाम है। याद रखने लायक मुख्य विचार पर भरोसा अधिक है: जो कोड चित्र बनाता है, उसके लिए चलते-चलते देखना अंधे होकर पूरा चित्र बनाने से बेहतर है।

स्रोत

Liang et al., Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback, arXiv:2604.20730 (2026)
`https://arxiv.org/abs/2604.20730`

OmniSVG project page
`https://omnisvg.github.io/`

InternSVG project page
`https://hmwang2002.github.io/release/internsvg/`