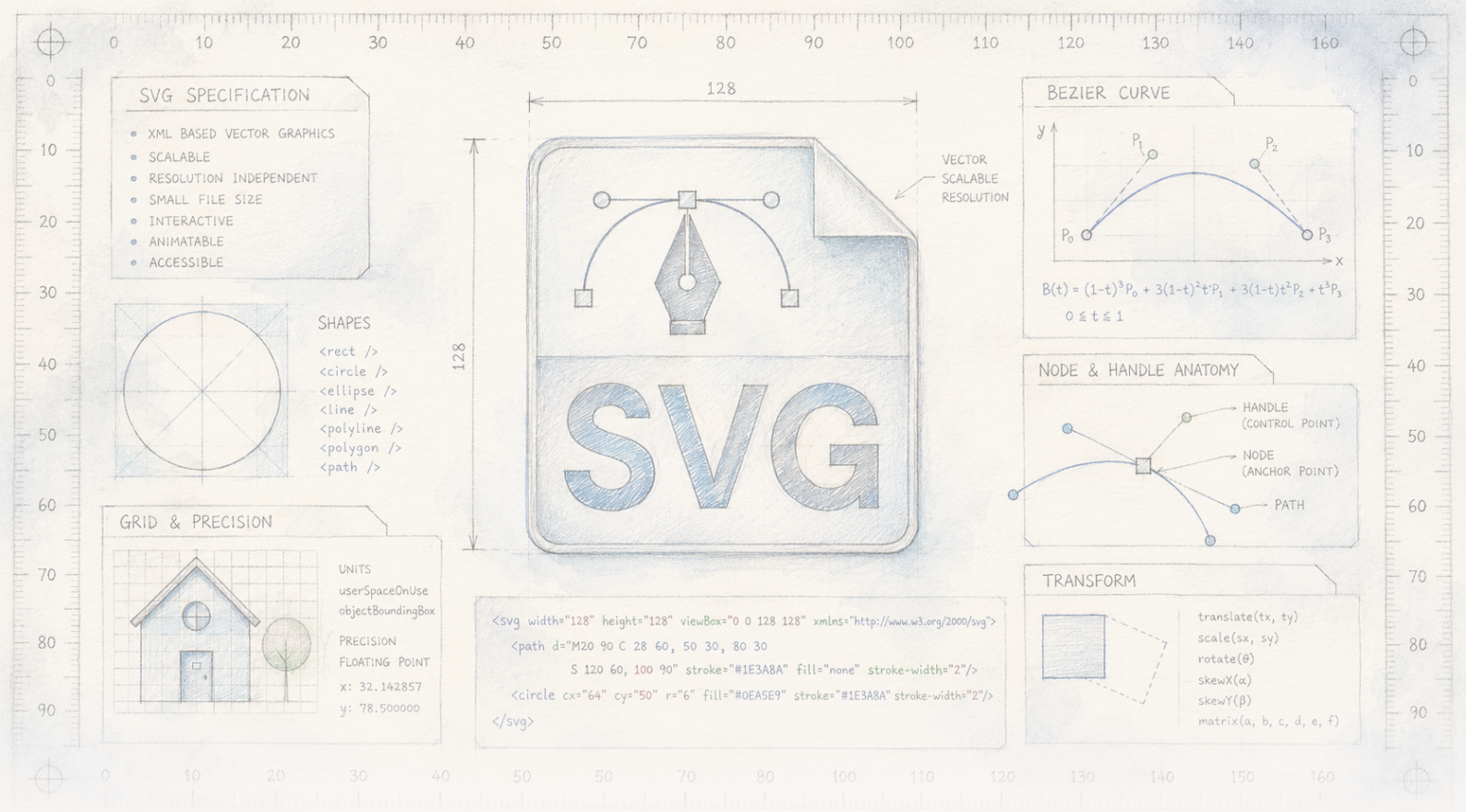


The Clean Paper

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

Dạy AI vẽ biết nhìn vào trang giấy

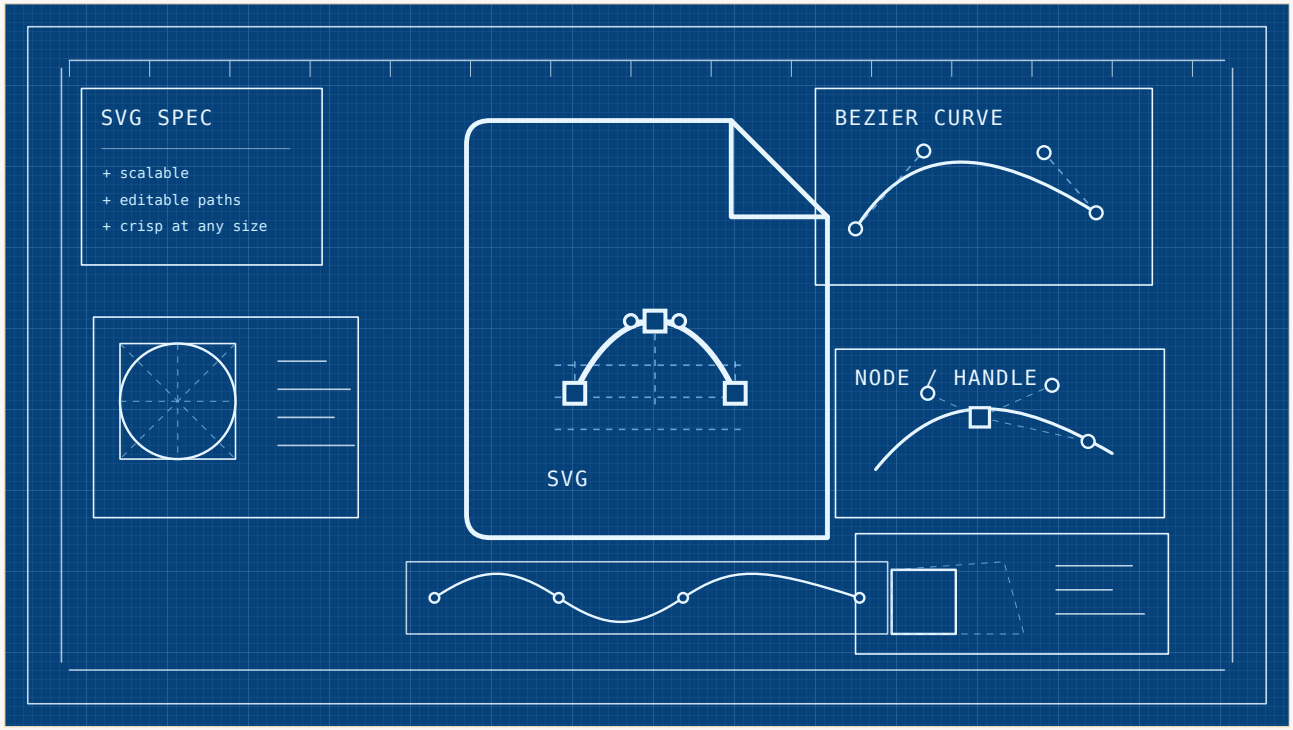
Các mô hình ngôn ngữ tạo đồ họa vector làm việc như bị bịt mắt — viết lệnh vẽ mà không bao giờ nhìn kết quả. Một phương pháp mới cho mô hình xem canvas của chính nó đầy lên từng nét, và điều thú vị trung thực là chỉ "cho mắt" khiến nó tệ hơn: phải huấn luyện lại để biết dùng mắt. Kết quả là một mô hình ngang hoặc hơi vượt các đối thủ được huấn luyện với dữ liệu nhiều hơn đến hai mươi lần — một kết quả thật và cẩn thận trên một benchmark, không phải cuộc cách mạng.

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)

Vẽ với đôi mắt nhắm

Hãy yêu cầu một mô hình AI ngày nay vẽ cho bạn một bức hình, và bên dưới lớp vỏ sẽ xảy ra một trong hai việc rất khác nhau. Những bộ tạo ảnh quen thuộc — loại biến một câu thành bức ảnh — vẽ trực tiếp các pixel và có thể "nhìn" canvas trong lúc làm. Nhưng có một kiểu vẽ thứ hai, yên lặng hơn, trong đó mô hình không tô màu gì cả. Nó viết chỉ dẫn: *vẽ một hình tròn ở đây, một đường ở kia, tô hình này màu xanh*. Những chỉ dẫn đó là mã — cùng loại Scalable Vector Graphics, hay SVG, nằm phía sau phần lớn biểu tượng và logo trên web — và lợi thế rất thật: kết quả không phải lưới pixel cố định mà là tập hợp hình có thể phóng to, đổi màu và chỉnh sửa mãi mà không bị mờ.



Bản vẽ kỹ thuật SVG gốc: bản thân hình là một tệp vector có thể chỉnh sửa, được dựng từ path, đường lưới, node và tay nắm thay vì các pixel cố định.

Laura Nesso / The Clean Paper Permission on file

Vấn đề là cho đến gần đây, mô hình viết mã vẽ này làm việc như bị bịt mắt. Nó xuất toàn bộ chuỗi lệnh trong một lượt — hình tròn, đường thẳng, tô màu — mà không bao giờ render để nhìn xem mình đã tạo gì. Hãy tưởng tượng phác một khuôn mặt khi nhắm mắt: bạn có thể đặt mắt gần đúng vị trí, nhưng không có cách nhận ra con mắt thứ hai đã rơi xuống má hay khối tóc vừa che mất mũi. Các mô hình này gần như làm vậy, nên chúng thường tạo mã hoàn toàn hợp lệ mà hình ảnh lại lộn xộn.

Một nhóm do Guotao Liang dẫn đầu giờ làm điều hiển nhiên đến mức gần như hài hước: cho mô hình mở mắt. Phương pháp *Render-in-the-Loop* render bản vẽ đang dở sau mỗi bước rồi đưa bức ảnh đó trở lại mô hình trước nét tiếp theo. Điều thực sự thú vị không phải "nhìn thì giúp" — mà là họ phải làm gì để việc nhìn có thể giúp được.

Chấm điểm một bức vẽ như thế nào?

Khi một bài báo nói mô hình của mình "thắng" mô hình khác, hoàn toàn hợp lý khi hỏi: thắng ở việc gì, đo bằng cách nào? Chấm một hình được sinh ra thật sự khó — không có một đáp án duy nhất cho "hãy vẽ laptop" — nên lĩnh vực dựa vào một số điểm tự động, mỗi điểm chỉ là đại diện không hoàn hảo cho việc con người nhìn kết quả.
Một vài chỉ số xuất hiện trong bài này. **FID** so sánh "hương vị thống kê" tổng thể của cả một lô ảnh sinh với ảnh thật; thấp hơn là tốt hơn, nhưng nó mô tả cả lô chứ không phải một ảnh. **CLIP score** hỏi một AI khác xem ảnh có khớp lời prompt không — hữu ích, nhưng chỉ sắc bằng chính giám khảo đó. **DINO**, **SSIM** và **LPIPS** so sánh ảnh tái dựng với ảnh mục tiêu ở các mức từ pixel thô đến đặc trưng đã học.
Không chỉ số nào là chân lý. Chúng là proxy và thường chỉ dịch chuyển từng chút. Khi đọc một mô hình đạt 127,6 còn mô hình khác 128,8, đó là khác biệt thật theo hướng mong muốn — nhưng chỉ là một cú nhích, không phải chiến thắng áp đảo, và là cú nhích trong một con số chỉ liên hệ lỏng lẻo với điều mắt người sẽ nói. Nên nhớ điều đó khi tiêu đề là "thắng mô hình được huấn luyện với dữ liệu gấp hai mươi lần".

Các tác giả đã làm gì

Vấn đề họ muốn sửa chính là việc vẽ mù. Các mô hình hiện có coi viết SVG như nhiệm vụ văn bản thuần túy: dự đoán đoạn mã tiếp theo từ mã đã có, không bao giờ render. Như vậy bỏ phí "đôi mắt" mạnh — bộ mã hóa thị giác — mà các mô hình đa phương thức hiện đại vốn đã mang. Render-in-the-Loop cấu trúc lại công việc thành một quá trình thị giác từng bước. Sau mỗi mảnh mã vẽ, SVG dở dang được render thành ảnh rồi đưa lại cho mô hình, để đoạn tiếp theo được chọn trong khi nó thực sự nhìn canvas hiện tại.

Phát hiện đầu tiên là một cảnh báo, và đáng khen là họ nói thẳng: chỉ gắn vòng lặp này vào một mô hình có sẵn không hiệu quả. Khi đưa các bản render trung gian cho những mô hình tổng quát mạnh mà không huấn luyện chuyên biệt, chất lượng không tăng — nó *giảm* ở mọi mặt. Một mô hình chưa từng học cách dùng mắt cho nhiệm vụ này không tự nhiên biết nhìn chỉ vì được cung cấp ảnh.

Vì vậy phần lớn công việc nằm ở việc dạy. Họ xây lại dữ liệu huấn luyện để mỗi hình được tách thành nhiều bước nhỏ có ý nghĩa thị giác — chia các hình phức tạp thành phần đơn giản để ở mỗi giai đoạn có thứ mới để nhìn — rồi fine-tune một mô hình mở tám tỷ tham số (dựa trên Qwen3-VL) trên các chuỗi từng bước đó. Họ gọi phần này là Visual Self-Feedback. Họ thêm cơ chế thứ hai lúc vẽ, Render-and-Verify: trước khi chấp nhận mỗi nét mới, mô hình render nó và kiểm tra xem nét đó thực sự thay đổi hình hay chỉ lặp lại nét trước. Nét không thêm gì bị bỏ; khi không còn gì hữu ích, mô hình được yêu cầu dừng. Đáng chú ý, tất cả chạy trên bộ dữ liệu khá nhỏ — khoảng 850.000 ví dụ, chưa bằng một nửa dữ liệu của một đối thủ và chỉ là phần nhỏ so với đối thủ khác.

Họ tìm thấy gì

- Khi được huấn luyện theo cách này, mô hình vẽ tốt hơn phiên bản "mù" — rõ nhất ở ca lỗi, nơi mô hình mù đặt mắt lên má hoặc bỏ qua biểu đồ cột được yêu cầu rồi xuất một màn hình chung chung.
- Trên benchmark chuẩn MMSVGBench, phương pháp cạnh tranh được và ở một số chỉ số nhỉnh hơn nhẹ các đối thủ mạnh — gồm OmniSVG, huấn luyện với dữ liệu hơn gấp đôi, và InternSVG, với dữ liệu khoảng gấp hai mươi lần.
- Biên thắng nhỏ. Chẳng hạn trên tập icon, điểm chất lượng ảnh chính là 127,6 so với 128,8 của đối thủ tốt nhất, còn điểm khớp prompt là 0,293 so với 0,291 — thật và nhất quán, nhưng mảnh.
- Cả hai thành phần thêm vào đều có tác dụng: tắt huấn luyện đặc biệt hoặc tắt bước kiểm tra lúc vẽ thì điểm giảm đo được. Bước kiểm tra đặc biệt giúp mô hình không mắc kẹt trong việc vẽ lại cùng một thứ hết lần này đến lần khác.
- Kết quả mà chính tác giả nhấn mạnh là hiệu quả dữ liệu — tiến đến mức này với dữ liệu huấn luyện ít hơn rất nhiều so với những mô hình dẫn đầu.

Điều nghiên cứu này không chứng minh

- Nó **không** cho thấy cho mô hình "nhìn" là một chiến thắng miễn phí. Thực tế ngược lại: thí nghiệm của bài cho thấy phản hồi thị giác *không kèm huấn luyện lại* làm kết quả tệ hơn. Lợi ích đến từ việc huấn luyện lại, không phải đôi mắt tự thân.
- Nó **không** thiết lập lợi thế lớn hay quyết định. Ở phần lớn điểm, phương pháp ngang ngửa đối thủ; "thắng mô hình được huấn luyện với dữ liệu gấp 20×" là đúng, nhưng chỉ bằng những cú nhích trên các chỉ số proxy và trên một benchmark.
- Nó **không** chứng minh năng lực nghệ thuật tổng quát. Đây là mô hình nghiên cứu 8 tỷ tham số vẽ icon và minh họa đơn giản ở độ phân giải cố định nhỏ (224×224 pixel), không phải nhà thiết kế đa dụng.
- So sánh với mô hình tổng quát lớn (GPT-5) **không** phải so táo với táo: mô hình đó không được xây hoặc tinh chỉnh cho nhiệm vụ hẹp "vẽ bằng mã", nên thắng ở đây nói rất ít về hai mô hình nói chung.
- Nó **không** miễn phí về tính toán. Render và đọc lại canvas ở mỗi bước khiến sinh chậm hơn việc xuất toàn bộ mã trong một lượt mù — chi phí mà tác giả thừa nhận.

Bằng chứng mạnh đến đâu

- **Vững** ở những so sánh có kiểm soát theo chính điều kiện của nghiên cứu. Các ablation rõ ràng: bỏ huấn luyện hoặc bỏ kiểm tra thì điểm giảm — vậy hai thành phần thật sự làm công việc tác giả nói.
- **Trung thực với điều bất ngờ của chính mình.** Phát hiện phản hồi thị giác ngây thơ *làm*

hại được báo cáo chứ không chôn đi, và đây có lẽ là điểm thú vị nhất: một sửa sai hữu ích cho trực giác "thêm đầu vào luôn tốt hơn".

- **Mỏng khi biến thành tuyên bố leaderboard.** Lợi thế trước đối thủ dùng nhiều dữ liệu là nhỏ và chỉ sống trên một benchmark do tác giả của một đối thủ góp phần xây. Biên nhỏ trên proxy metrics ở một test set là gợi ý, chưa phải phán quyết.
- **Chưa thử ở quy mô và ngoài đời.** Mọi thứ ở đây là icon và minh họa đơn giản độ phân giải thấp. Ý tưởng có giữ được với đồ họa phức tạp, độ phân giải cao hay công việc thiết kế thực tế không vẫn là việc tương lai — chính tác giả nói vậy.

Vì sao điều này quan trọng

Ý tưởng trung tâm đơn giản đến gần như ngượng ngùng, và vượt xa chuyện vẽ. Nếu một chương trình tạo thứ gì đó bằng cách viết mã — trang web, biểu đồ, sơ đồ, cảnh 3D — nó có thể viết toàn bộ trong mù rồi hy vọng, hoặc render trong lúc làm và sửa hướng. Khép vòng lặp này là bản năng thứ hai với con người; chúng ta liên tục liếc trang giấy. Với các mô hình, đây lại là động tác khá mới.

Điều khiến bài đáng đọc là dấu hoa thị nó gắn vào ý tưởng. Cho mô hình cách nhìn không giống với dạy nó *biết nhìn*. Đôi mắt phải được huấn luyện, rồi vòng lặp mới có lợi — và ngay cả khi đó, lợi ích thật nhưng có chừng mực: một người vẽ ổn định hơn, không phải một loại nghệ sĩ khác. Với ai đang xây công cụ biến mô tả thành đồ họa có thể chỉnh sửa — loại thứ nằm phía sau icon và minh họa của ứng dụng — bài học thực tế yên lặng mới hữu ích. Chiến thắng ở đây đến từ huấn luyện rẻ hơn, thông minh hơn, không phải nhiều dữ liệu hơn. Đó là điều đáng học hơn một điểm leaderboard nữa.

Tóm tắt gọn

Các mô hình ngôn ngữ tạo đồ họa vector — mã có thể chỉnh sửa và phóng to nằm sau phần lớn icon và logo web — từ trước đến nay thường làm việc "mù", viết toàn bộ lệnh vẽ mà không render để nhìn kết quả. Nhóm của Guotao Liang đề xuất Render-in-the-Loop: render hình đang dở sau mỗi bước rồi đưa lại, để mô hình vẽ nét tiếp theo trong khi nhìn canvas. Phát hiện trung tâm và trung thực là chỉ làm vậy với mô hình hiện có khiến nó *tệ hơn*; lợi ích chỉ xuất hiện sau khi mô hình được huấn luyện lại để dùng phản hồi thị giác, cùng một bước kiểm tra lúc vẽ loại bỏ những nét không thay đổi gì. Mô hình 8 tỷ tham số đã huấn luyện lại ngang hoặc hơi vượt các đối thủ dùng dữ liệu nhiều hơn đến hai mươi lần trên benchmark chuẩn — kết quả thật, nhưng biên nhỏ trên chỉ số proxy, với icon và minh họa đơn giản độ phân giải thấp. Điều tác giả nhấn mạnh, và đáng giữ lại, là hiệu quả: nhìn tốt sản phẩm đang làm có thể bù cho việc chỉ tăng quy mô dữ liệu.

Đánh giá thẳng thắn

Bài báo cho thấy gì: Huấn luyện lại một mô hình đồ họa vector để render và nhìn hình đang dở từng bước tạo kết quả tốt và đầy đủ hơn vẽ mù — cạnh tranh được, hơi nhỉnh hơn đối thủ dùng nhiều dữ liệu trên một benchmark chuẩn, trong khi dùng ít dữ liệu huấn luyện hơn.

Điều hợp lý nhưng chưa được chứng minh: "Nhìn sản phẩm đang làm" rộng rãi là công thức tốt hơn tăng dữ liệu; cùng ý tưởng sẽ giúp đồ họa phức tạp, độ phân giải cao hay công việc ngoài đời; các biên benchmark nhỏ phản ánh khác biệt mà con người thực sự nhận thấy.

Điều nghiên cứu không cho thấy: Phản hồi thị giác tự thân có ích (không huấn luyện lại thì nó làm hại); lợi thế trước đối thủ lớn hay quyết định; khả năng vẽ đa dụng; so sánh công bằng trực tiếp với mô hình tổng quát như GPT-5 vốn không được xây cho nhiệm vụ này.

Hạn chế chính: Một benchmark, phần nào do tác giả của đối thủ xây; biên nhỏ trên proxy metrics; mô hình 8B, icon và minh họa đơn giản ở 224×224; sinh chậm hơn vì vòng lặp render mỗi bước.

Độc giả phổ thông nên tin ở mức nào? Cao rằng khép vòng lặp — render và đọc lại trong lúc vẽ — thực sự giúp và phải được huấn luyện chứ không chỉ bật lên. Thấp đến trung bình rằng mô hình cụ thể này thắng đối thủ một cách quyết định; hãy đọc câu "thắng với 20× ít dữ liệu hơn" như một kết quả thật nhưng khiêm tốn trên một benchmark. Cao với ý tưởng đáng nhớ: với mã dùng để vẽ, nhìn trong lúc làm tốt hơn vẽ mù.

Nguồn

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/