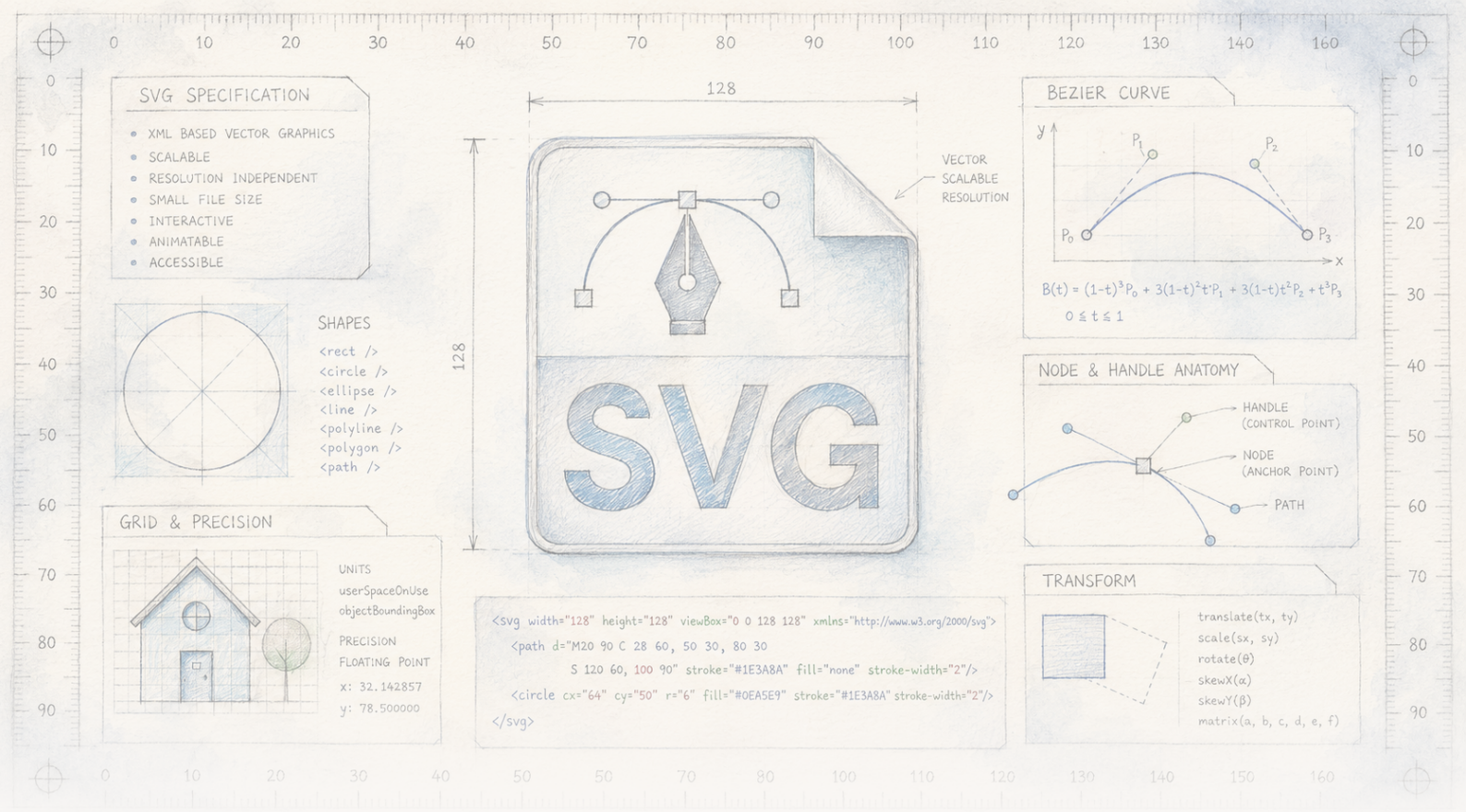


The Clean Paper

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

Insegnare a una AI che disegna a guardare la pagina

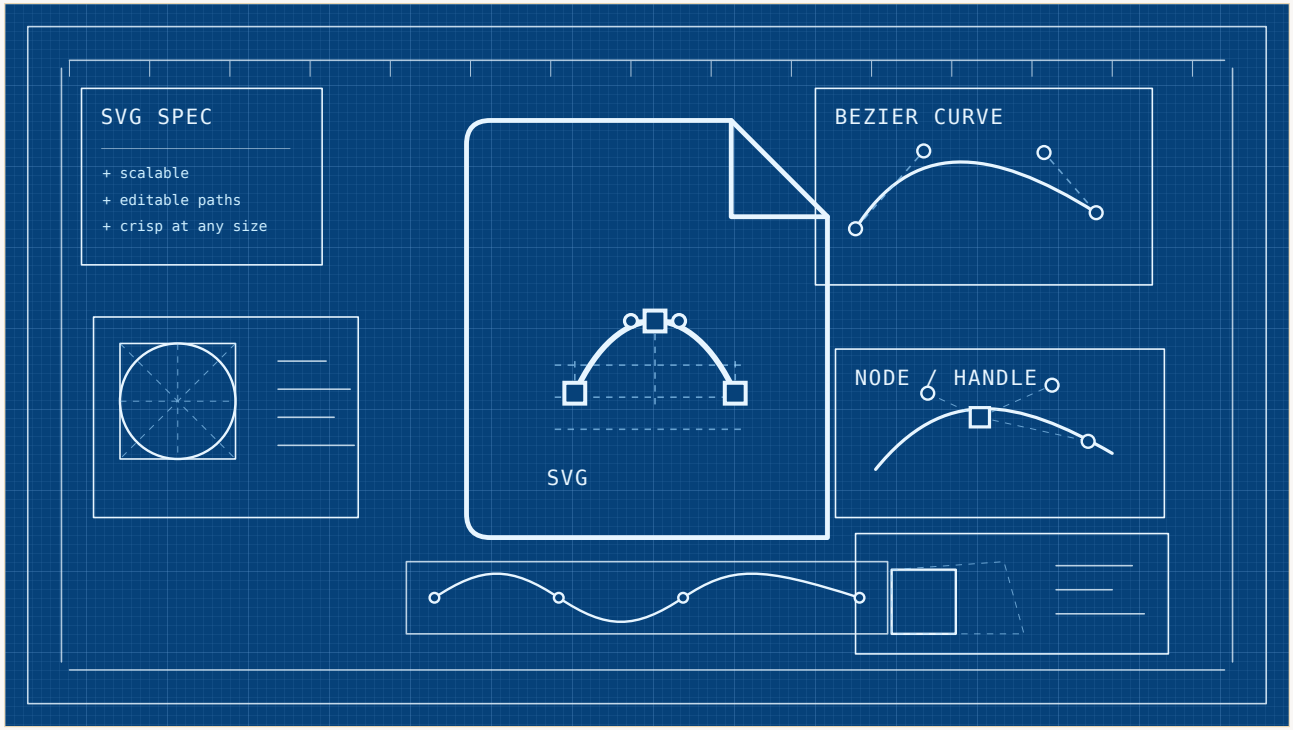
I modelli linguistici che generano grafica vettoriale lo fanno alla cieca: scrivono i comandi del disegno senza vedere il risultato. Un nuovo metodo lascia che il modello guardi la propria tela riempirsi, tratto dopo tratto, e la svolta onesta è che dargli semplicemente gli occhi peggiora le cose: deve essere riaddestrato per usarli. Il risultato è un modello che eguaglia o supera di poco rivali addestrati con fino a venti volte più dati — un risultato reale e cauto su un solo benchmark, non una rivoluzione.

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)

Disegnare a occhi chiusi

Chiedi a uno dei modelli AI di oggi di disegnarti un'immagine e, sotto il cofano, succede una di due cose molto diverse. I generatori di immagini familiari — quelli che evocano una fotografia da una frase — dipingono direttamente pixel, e possono vedere la tela mentre lavorano. Ma c'è un secondo tipo di disegno, più silenzioso, in cui il modello non dipinge affatto. Scrive istruzioni: *disegna un cerchio qui, una linea lì, riempi questa forma di blu*. Quelle istruzioni sono codice — lo stesso Scalable Vector Graphics, o SVG, che sta dietro la maggior parte delle icone e dei loghi sul web — e il vantaggio è reale: il risultato non è una griglia fissa di pixel, ma un insieme di forme che puoi scalare, ricolorare e modificare per sempre senza farle sfocare.



Artwork originale in forma di blueprint SVG: l'immagine è essa stessa un file vettoriale modificabile, costruito da tracciati, linee di griglia, nodi e maniglie invece che da pixel fissi.

Laura Nesso / The Clean Paper Permission on file

Il problema è che, fino a poco tempo fa, un modello che scriveva questo codice di disegno lo faceva alla cieca. Emetteva l'intera sequenza di istruzioni in un solo passaggio — cerchio, linea, riempimento — senza mai renderizzarle per vedere cosa aveva prodotto. Immagina di disegnare un volto a occhi chiusi: potresti mettere gli occhi più o meno dove vanno gli occhi, ma non avresti modo di accorgerti che il secondo è finito sulla guancia, o che la forma disegnata per i capelli ora copre il naso. Più o meno così lavoravano questi modelli, ed è per questo che così spesso producevano codice perfettamente valido e insieme un disastro visivo.

Un team guidato da Guotao Liang ha fatto ora la cosa quasi comicamente ovvia: ha lasciato che il modello aprisse gli occhi. Il loro metodo, *Render-in-the-Loop*, renderizza il disegno a metà dopo ogni passo e restituisce quell'immagine al modello prima che disegni il tratto successivo. La parte davvero

interessante non è che questo aiuti — è ciò che hanno dovuto fare perché aiutasse davvero.

Come si dà un voto a un disegno?

Quando un lavoro dice che il suo modello "batte" un altro, è giusto chiedere: lo batte in cosa, e misurato come? Valutare un'immagine generata è davvero difficile — non esiste una risposta unica e corretta a "disegna un laptop" — quindi il campo si appoggia a una manciata di punteggi automatici, ciascuno un sostituto imperfetto di una persona che guarda il risultato.
Alcuni compaiono in questo lavoro. **FID** confronta il sapore statistico complessivo di un intero lotto di immagini generate con immagini reali; più basso è meglio, ma descrive il lotto, non una singola figura. **CLIP score** chiede a un'altra AI se un'immagine corrisponde alle parole del prompt — utile, ma affilato solo quanto quel giudice. **DINO**, **SSIM** e **LPIPS** confrontano un'immagine ricostruita con un bersaglio, a livelli che vanno dai pixel grezzi a caratteristiche apprese.
Nessuno di questi è verità. Sono proxy, e tendono a muoversi per piccoli incrementi. Quando leggi che un modello fa 127,6 dove un altro fa 128,8, è una differenza reale nella direzione voluta — ma è una spinta leggera, non una frana, e una spinta in un numero che segue solo vagamente ciò che direbbe il tuo occhio. Vale la pena tenerlo a mente quando il titolo è "batte un modello addestrato con venti volte più dati".

Cosa hanno fatto gli autori

Il problema che gli autori volevano risolvere è il disegno alla cieca. I modelli esistenti trattano la scrittura di SVG come un puro compito testuale: prevedi il prossimo pezzo di codice dal codice già scritto, e non renderizzare mai. Questo lascia inutilizzati gli "occhi" potenti — l'encoder visivo — che i modelli multimodali moderni già portano con sé. Render-in-the-Loop ristruttura il lavoro come un processo visivo passo per passo. Dopo ogni frammento di codice di disegno, l'SVG parziale viene renderizzato in un'immagine e restituito al modello come figura, così il frammento successivo viene scelto guardando davvero la tela fino a quel momento.

Il primo risultato è un avvertimento, e gli autori lo riportano con chiarezza: applicare semplicemente questo loop a un modello esistente, già pronto, non funziona. Quando hanno dato render intermedi a modelli generali forti senza alcun addestramento speciale, la qualità non è migliorata — è *peggiorata* su tutta la linea. Un modello a cui non è mai stato insegnato a usare gli occhi per questo non impara all'improvviso come farlo.

Quindi gran parte del lavoro è nell'insegnamento. Ricostruiscono i dati di addestramento in modo che ogni disegno sia spezzato in molti passi piccoli e visivamente significativi — dividendo forme complesse in pezzi più semplici, così a ogni stadio c'è qualcosa di nuovo da vedere — e poi fanno fine-tuning di un modello aperto da otto miliardi di parametri (basato su Qwen3-VL) su queste sequenze passo per passo. Lo chiamano Visual Self-Feedback. Aggiungono un secondo meccanismo al momento del disegno, Render-and-Verify: prima di accettare ogni nuovo tratto, il modello lo renderizza e controlla se ha davvero cambiato l'immagine, o se ha solo ripetuto l'ultimo. I tratti che non aggiungono nulla

vengono scartati, e quando nulla aiuta più, il modello viene spinto a fermarsi. Nota importante: tutto questo gira su un dataset piuttosto piccolo — circa 850.000 esempi, meno della metà di quanto ha usato un rivale e una piccola frazione di quello di un altro.

Cosa hanno trovato

- Addestrato così, il modello disegna meglio della sua controparte cieca — soprattutto nei casi di fallimento, dove un modello cieco mette un occhio su una guancia, o salta un grafico a barre richiesto e produce invece un monitor generico.
- Sul benchmark standard (MMSVGBench), il metodo risulta competitivo con rivali forti, e su alcune misure leggermente avanti — inclusi OmniSVG, addestrato con più del doppio dei dati, e InternSVG, addestrato con circa venti volte tanto.
- I margini sono piccoli. Sul set di icone, per esempio, il suo punteggio principale di qualità dell'immagine è 127,6 contro 128,8 del miglior rivale, e il punteggio di corrispondenza col prompt 0,293 contro 0,291 — reale e coerente, ma sottile.
- Entrambi i pezzi aggiunti servono: togli l'addestramento speciale, o togli la verifica al momento del disegno, e i numeri calano in modo misurabile. La verifica in particolare impedisce al modello di restare bloccato a ridisegnare la stessa cosa ancora e ancora.
- Il risultato che gli autori stessi enfatizzano è l'efficienza — arrivare fin qui con molti meno dati di addestramento di quelli usati dai leader.

Cosa non dimostra

- **Non** mostra che lasciare che un modello "veda" sia una vittoria gratis. Anzi: l'esperimento del lavoro mostra che il feedback visivo *senza* riaddestramento peggiora le cose. Il guadagno viene dal riaddestramento, non dagli occhi da soli.
- **Non** stabilisce un vantaggio grande o decisivo. Sulla maggior parte dei punteggi il metodo è testa a testa con i rivali; "batte un modello addestrato con 20× i dati" è vero, ma per piccole spinte su metriche proxy, su un solo benchmark.
- **Non** dimostra capacità artistica generale. Questo è un modello di ricerca da otto miliardi di parametri che disegna icone e illustrazioni semplici a una piccola risoluzione fissa (224×224 pixel), non un designer generalista.
- Il confronto con un grande modello generale (GPT-5) **non** è apples-to-apples: quel modello non è costruito né ottimizzato per questo compito stretto di disegno via codice, quindi batterlo qui dice poco su entrambi i modelli in generale.
- **Non** arriva gratis. Renderizzare e rileggere la tela a ogni passo rende la generazione più lenta che emettere il codice in un solo passaggio cieco — un costo che gli autori riconoscono.

Quanto sono solide le prove?

- **Solida** dove è un confronto controllato sui suoi stessi termini. Le ablation sono pulite: togli l'addestramento, o togli la verifica, e i numeri scendono — quindi i due ingredienti fanno davvero il lavoro che gli autori attribuiscono loro.
- **Onesta sulla propria sorpresa.** Il fatto che il feedback visivo ingenuo *faccia male* viene riportato, non sepolto, ed è la cosa più interessante del lavoro — una correzione utile all'intuizione che più input sia sempre meglio.
- **Sottile dove diventa una classifica.** Le vittorie sui rivali addestrati con più dati sono piccole e vivono su un solo benchmark costruito dagli autori di uno di quei rivali. Piccoli margini su metriche proxy in un solo test set sono suggestivi, non definitivi.
- **Non testata su scala e nel mondo reale.** Qui tutto riguarda icone e illustrazioni semplici a bassa risoluzione. Se la stessa idea regga per grafica complessa, ad alta risoluzione o per lavoro di design reale resta lavoro futuro — lo dicono gli autori stessi.

Perché conta

L'idea al centro di questo lavoro è quasi imbarazzante nella sua semplicità, e va ben oltre il disegno. Se un programma deve generare qualcosa scrivendo codice — una pagina web, un grafico, un diagramma, una scena 3D — può scrivere tutto alla cieca e sperare, oppure renderizzare mentre procede e correggere la rotta. Chiudere quel loop è naturale per una persona: guardiamo continuamente la pagina. Per questi modelli, è una mossa sorprendentemente recente.

Ciò che rende il lavoro degno di lettura è l'asterisco che attacca a questa idea. Dare a un modello un modo per vedere non è lo stesso che insegnargli a guardare. Gli occhi devono essere addestrati, e solo allora il loop ripaga — e anche allora il risultato è reale ma misurato: un disegnatore più stabile, non un artista di un altro tipo. Per chi costruisce strumenti che trasformano una descrizione in una grafica modificabile — il genere di cosa che finisce dietro icone e illustrazioni di un'app — la lezione utile è quella quieta e pratica. La vittoria qui è arrivata da un addestramento più economico e più intelligente, non da più dati. È una cosa migliore da imparare di un altro punto su una classifica.

In breve

I modelli linguistici che generano grafica vettoriale — il codice modificabile e scalabile dietro la maggior parte delle icone e dei loghi del web — tradizionalmente lo fanno "alla cieca", scrivendo tutti i comandi del disegno senza mai renderizzarli per vedere il risultato. Un team guidato da Guotao Liang propone Render-in-the-Loop: renderizzare il disegno a metà dopo ogni passo e restituirlo al modello, così il tratto successivo viene disegnato guardando la tela. Il loro risultato centrale e onesto è che farlo semplicemente a un modello esistente lo rende *peggiore*; il guadagno appare solo dopo aver riaddestrato il modello a usare il feedback visivo, aiutato da un controllo al momento del disegno che

scarta i tratti che non cambiano nulla. Il modello riaddestrato da otto miliardi di parametri eguaglia o supera di poco rivali addestrati con fino a venti volte più dati su un benchmark standard — un risultato genuino, ma con margini piccoli su punteggi proxy, per icone e illustrazioni semplici a bassa risoluzione. Il punto che gli autori sottolineano, e quello da conservare, riguarda l'efficienza: vedere bene il proprio lavoro può sostituire in parte la pura scala dei dati.

Valutazione critica

Cosa mostra il lavoro: Riaddestrare un modello di grafica vettoriale perché renderizzi e guardi il proprio disegno a metà, passo dopo passo, produce risultati migliori e più completi del disegno alla cieca — competitivi con, e leggermente davanti a, rivali addestrati con più dati su un benchmark standard, usando meno dati di addestramento.

Cosa è plausibile ma non provato: Che "vedere il proprio lavoro" sia in generale una ricetta migliore che scalare i dati; che la stessa idea aiuti su grafica complessa, ad alta risoluzione o nel mondo reale; che i piccoli margini del benchmark riflettano una differenza che una persona noterebbe davvero.

Cosa non mostra: Che il feedback visivo aiuti da solo (senza riaddestramento fa male); che il vantaggio sui rivali sia grande o decisivo; capacità di disegno generalista; un confronto diretto corretto con modelli generali come GPT-5, che non sono costruiti per questo compito.

Limiti principali: Un solo benchmark, costruito in parte dagli autori di un rivale; piccoli margini su metriche proxy; un modello da 8B, icone e illustrazioni semplici a 224×224; generazione più lenta per via del loop che renderizza a ogni passo.

Quanta fiducia dovrebbe avere un lettore generale? Alta sul fatto che chiudere il loop — renderizzare e rileggere mentre si disegna — aiuti davvero, e che debba essere addestrato, non solo acceso. Bassa-moderata sul fatto che questo modello specifico batta decisamente i rivali; tratta la frase "batte 20× i dati" come un risultato reale ma modesto, su un solo benchmark. Alta sull'idea da ricordare: per il codice che disegna, guardare mentre procedi batte disegnare alla cieca.

Fonti

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/