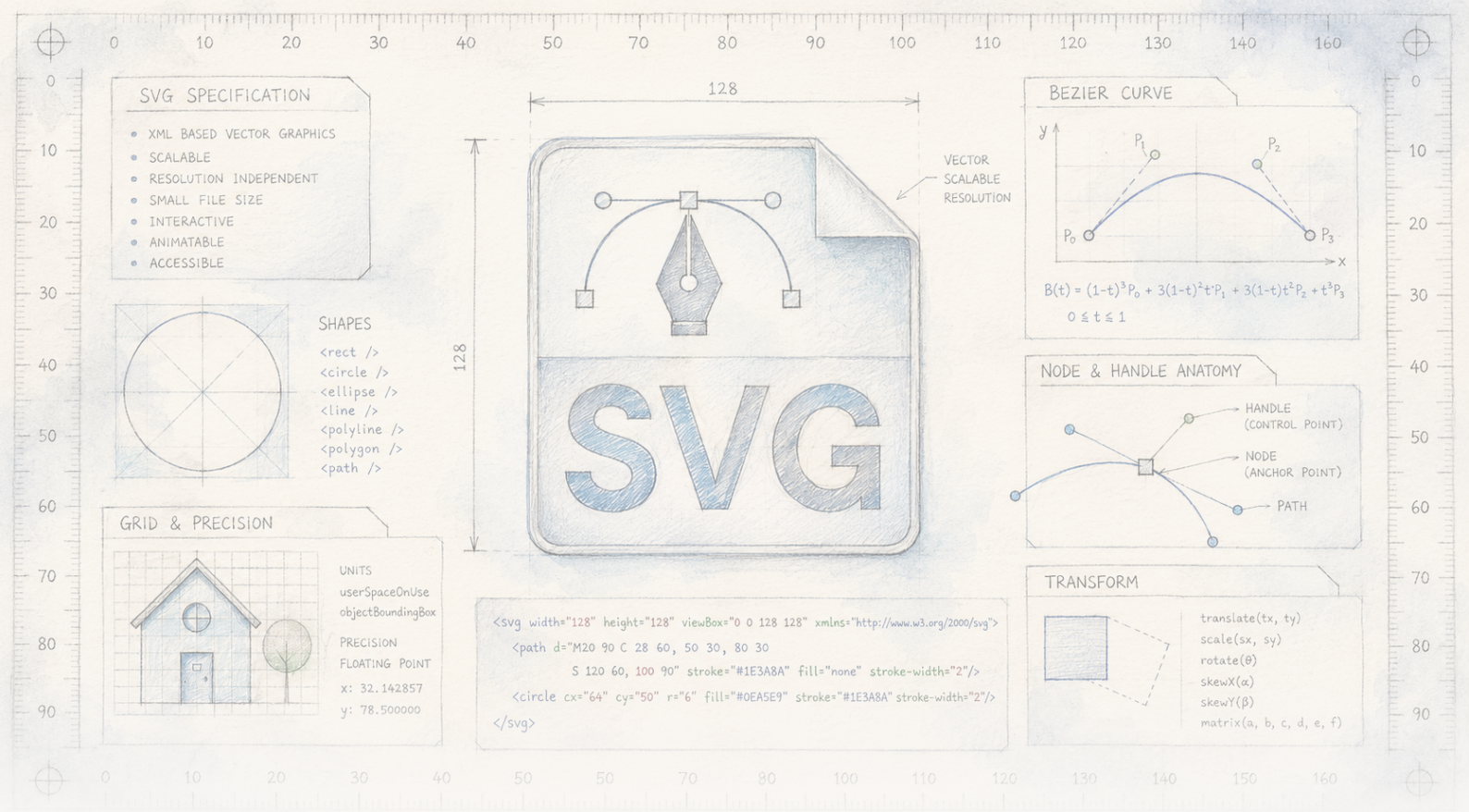



The Clean Paper

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

# 教繪圖 AI 看著畫布作畫

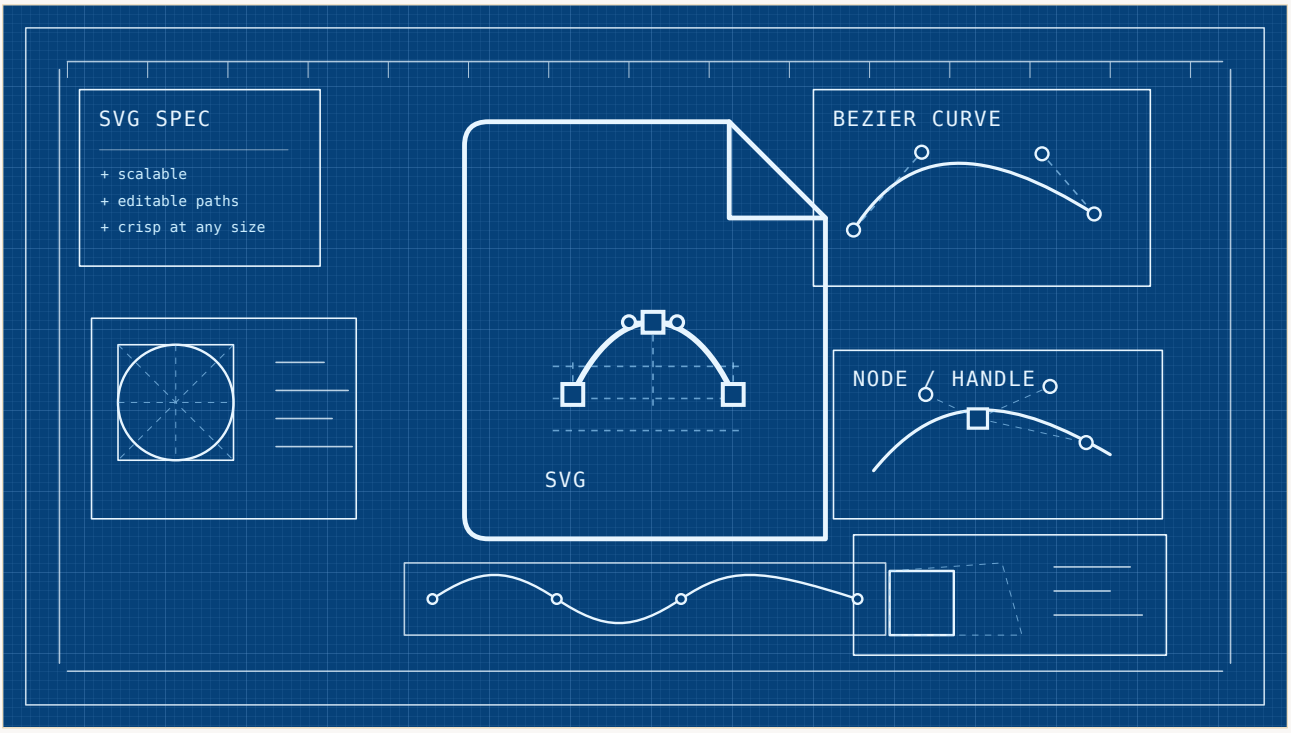
生成向量圖形的語言模型一直在盲畫——寫出繪圖命令，卻從未看見結果。一種新方法讓模型逐筆看著自己的畫布被填滿，而誠實的轉折是，簡單地賦予它眼睛反而讓結果變差：模型必須經過再訓練，學會怎樣使用視覺回饋。回報是一個能追平或略微勝過使用最多二十倍資料訓練的對手的模型——這是在單一基準上得到的真實、謹慎的結果，不是一場革命。

Written by Vera Rubin · Cross-checked by Michele Renda · Edited by Michele Renda · Figures by Laura Nesso · 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)

# 閉著眼睛畫畫

讓當今的一個 AI 模型為你畫張圖，底層會發生兩種截然不同的事情之一。人們熟悉的影像生成器——那些能從一句話變出一張照片的模型——直接繪製像素，而且工作時看得見畫布。但還有第二種更安靜的繪圖方式，模型根本不直接作畫，而是編寫指令：在這裡畫個圓，在那裡畫條線，把這個形狀填成藍色。這些指令是程式碼——也就是支撐網路上大多數圖示和標誌的可縮放向量圖形（SVG）——它的吸引力很真實：結果不是固定的像素網格，而是一組可以無限縮放、改色和編輯、卻永遠不會變模糊的形狀。




原創 SVG 藍圖作品：影像本身就是可編輯的向量檔案，由路徑、網格線、節點和控制柄構成，而不是固定像素。

Laura Nesso / The Clean Paper Permission on file

問題在於，直到最近，模型編寫這種繪圖程式碼時都是盲畫。它會一次性輸出整串指令——圓、線、填充——從不渲染出來看看自己畫了什麼。想像一下閉著眼睛畫一張臉：眼睛大概會落在該有的位置，但你無從發現第二隻眼睛畫到了臉頰上，也不會知道為頭髮畫的形狀此刻正蓋在鼻子上。這些模型過去的工作方式大致如此，因此它們經常生成完全有效、視覺上卻一團糟的程式碼。

梁國濤帶領的團隊如今做了一件近乎滑稽地顯而易見的事：讓模型睜開眼睛。他們的方法 *Render-in-the-Loop* 會在每一步之後渲染尚未完成的圖畫，並在模型落下下一筆之前把影像回饋給它。真正有意思的並不是這樣做有所幫助，而是他們必須怎樣做，才能讓它真正有所幫助。

### 怎樣給一幅畫評分？

當一篇論文說自己的模型「擊敗」了另一個模型時，追問一句很公平：在哪方面擊敗，又是怎樣測量的？判斷生成影像的品質確實很難——「畫一台筆記本電腦」沒有唯一正確答案——因此

這個領域依賴少數幾種自動評分，每一種都只是人眼觀察結果的不完美代理指標。
本文採用了其中幾種。**FID** 把一整批生成影像的總體統計風格與真實影像比較；數值越低越好，但它描述的是整批影像，而不是某一張。**CLIP 分數**讓另一個 AI 判斷影像是否符合提示中的文字——有用，但其敏銳程度不會超過這個評判者本身。**DINO**、**SSIM** 和 **LPIPS** 則在從原始像素到學習特徵的不同層面上，將重建影像與目標影像進行比較。
這些評分都不是真理，只是代理指標，而且變化通常很小。當你看到一個模型得 127.6 分，另一個得 128.8 分時，這確實是朝預期方向的真實差異——但它只是輕輕一推，不是壓倒性勝利，而且這個數字本身與人眼判斷的對應也很鬆散。看到「擊敗使用二十倍資料訓練的模型」這類標題時，值得記住這一點。

# 作者做了什麼

作者著手解決的問題就是這種盲畫。現有模型把編寫 SVG 視為純文字任務：根據已有程式碼預測下一段程式碼，從不進行渲染。這讓現代多模態模型本已具備的強大「眼睛」——視覺編碼器——閒置不用。Render-in-the-Loop 把任務重構成逐步進行的視覺過程。每生成一段繪圖程式碼，就把部分 SVG 渲染成影像，再以圖片形式回饋給模型，讓模型真正看著當前畫布來選擇下一段程式碼。

他們的第一個發現帶有警示意味，而且值得肯定的是，他們直言不諱地報告了這一點：僅僅把這個循環接到現成模型上並不起作用。研究人員在沒有專門訓練的情況下，把中間渲染結果輸入強大的通用模型，品質不但沒有提高，反而全面下降。一個從未學過怎樣用眼睛完成這項工作的模型，不會突然無師自通。

因此，大部分工作都在於教學。團隊重建訓練資料，把每幅畫拆分成許多細小而具有視覺意義的步驟——把複雜形狀拆成更簡單的部件，使每個階段都有新內容可看——然後使用這些分步序列，對一個基於 Qwen3-VL、擁有 80 億參數的開放模型進行微調。他們把這稱為視覺自回饋（Visual Self-Feedback）。繪圖時又加入第二套機制 Render-and-Verify：在接受每一筆之前，模型先將其渲染並檢查它是否真的改變了影像，還是只重複了上一筆。沒有增加任何內容的筆畫會被丟棄；當再也沒有什麼能改善影像時，模型會收到停止提示。值得注意的是，這一切只使用了相當小的資料集——約 850,000 個樣本，不到一個競爭對手所用資料的一半，也只是另一個對手所用資料的一小部分。

# 他們發現了什麼

- 經過這種訓練，模型比其盲畫版本表現更好——在失敗案例中最為明顯：盲畫模型會把眼睛畫在臉頰上，或跳過提示要求的條形圖，轉而輸出一個普通顯示器。
- 在標準基準 MMSVGBench 上，該方法與強勁對手相當，並在若干指標上略微領先——包括訓練資料超過其兩倍的OmniSVG，以及訓練資料約為其二十倍的InternSVG。
- 優勢幅度很小。例如在圖示資料集上，它的主要影像品質分數為 127.6，最佳對手為 128.8；它的提示詞匹配分數為 0.293，對手為 0.291——真實且一致，但很微弱。
- 兩個新增部分都不可或缺：關閉專門訓練或繪圖時驗證，分數都會出現可測量的下降。尤其是驗證步驟，可以阻止模型陷入反覆重畫同一內容的循環。

- 作者自己強調的結果是效率——用遠少於領先模型的資料，取得了這樣的表現。

# 這項研究沒有證明什麼

- 它**沒有**表明讓模型「看見」就能免費獲益。事實上恰恰相反：論文自己的實驗顯示，不經過再訓練的視覺回饋會讓結果更差。提升來自再訓練，而不是單靠眼睛。
- 它**沒有**確立巨大或決定性的領先。在大多數分數上，該方法與對手不相上下；「擊敗使用 20 倍資料訓練的模型」確實屬實，但只是在一個基準的代理指標上輕微領先。
- 它**沒有**展示通用藝術能力。這是一個 80 億參數的研究模型，以較小的固定解析度（224×224 像素）繪製圖示和簡單插圖，不是通用設計師。
- 與大型通用模型 GPT-5 的比較**並非同類比較**：該模型並不是為這種狹窄的程式碼繪圖任務建構或調校的，因此在這裡勝過它，並不能說明兩個模型的總體能力。
- 它**不是**沒有代價。在每一步都渲染並重新讀取畫布，會比一次性盲目輸出程式碼更慢——作者也承認這一成本。

# 證據有多強

- 在論文自身設定下進行受控比較的部分，證據**扎實**。消融實驗很乾淨：移除訓練或移除驗證，分數都會下降——因此這兩個組成部分確實在發揮作者所說的作用。
- 論文**誠實面對自己的意外發現**。樸素視覺回饋反而有害這一結果被報告出來，而不是藏起來；它也是論文中最有意思的一點——有力地修正了「輸入越多總是越好」的直覺。
- 涉及排行榜主張的部分則**較薄弱**。相對於使用更多資料的對手，領先幅度很小，而且只出現在單一基準上；這個基準還是由其中一個對手模型的作者建立的。在一個測試集的代理指標上小幅領先，具有提示性，卻不是定論。
- **尚未在大規模和真實環境中測試**。這裡的一切都限於低解析度圖示和簡單插圖。同一種思想是否適用於複雜、高解析度或真實世界的設計工作，仍留待未來研究——作者也明確這樣說。

# 為什麼重要

這篇論文的核心思想簡單得幾乎令人尷尬，卻遠遠超出繪圖本身。如果一個程式要透過編寫程式碼來生成東西——網頁、圖表、示意圖、3D 場景——它可以閉著眼睛把全部內容寫完再碰運氣，也可以邊渲染邊修正方向。對人而言，閉合這個回饋迴路是自然而然的；我們會不斷瞥一眼頁面。對這些模型來說，這卻是一個出乎意料地晚近的做法。

論文值得一讀，是因為它在這個想法後面加了一個星號。讓模型能夠看見，不等於教會它觀察。必須訓練這雙眼睛，回饋迴路才會帶來回報——即便如此，回報也真實但有限：得到的是一位更穩定的繪圖員，而不是一種全新的藝術家。對於任何建構「把描述變成可編輯圖形」工具的人——比如最終成為應用程式圖示和插圖的那類東西——真正有用的是這條安靜而實際的教訓。這裡的勝利來自更便宜、更聰明的訓練，而不是更多資料。這比排行榜上再多一個點更值得學習。

# 清晰摘要

生成向量圖形的語言模型——也就是生成大多數網路圖示和標誌背後可編輯、可縮放程式碼的模型——傳統上一直在「盲畫」：寫出全部繪圖命令，卻從不渲染出來查看結果。梁國濤帶領的團隊提出 Render-in-the-Loop：每一步之後渲染尚未完成的圖畫並回饋給模型，讓它看著畫布落下下一筆。他們最核心、也最坦誠的發現是，簡單地把這一做法應用到現有模型上，反而會讓模型變差；只有經過再訓練、學會利用視覺回饋，並輔以丟棄無效筆畫的繪圖時檢查，效能才會提升。再訓練後的 80 億參數模型在一項標準基準上追平或略微勝過使用最多二十倍資料訓練的對手——這是真實結果，但只在低解析度圖示和簡單插圖的代理指標上小幅領先。作者強調、也值得記住的結論關乎效率：善於觀察自己的作品，可以在一定程度上替代純粹的資料規模。

# 不繞彎核查

**論文證明了什麼：**對向量圖形模型進行再訓練，使其逐步渲染並觀察自己尚未完成的圖畫，會比盲畫產生更好、更完整的結果——在一項標準基準上，用更少的訓練資料達到與大資料對手相當、並略微領先的表現。

**什麼合理但尚未證明：**「觀察自己的作品」廣泛而言是比擴大資料規模更好的方案；同一種思想能幫助複雜、高解析度或真實世界的圖形任務；基準上的微小優勢反映了人能實際察覺的差異。

**它沒有證明什麼：**視覺回饋單獨使用就有幫助（不經再訓練反而有害）；對競爭模型的領先巨大或具有決定性；擁有通用繪圖能力；與 GPT-5 等並非為此任務建構的通用模型進行的是公平正面對比。

**主要限制：**只使用一個基準，且該基準部分由一個對手模型的作者建立；代理指標上的優勢很小；80 億參數模型，繪製 224×224 的圖示和簡單插圖；逐步渲染循環導致生成速度較慢。

**普通讀者應該有多大信心？**可以高度確信，閉合回饋迴路——邊畫邊渲染和重新觀察——確實有幫助，而且這種能力必須透過訓練獲得，不能簡單開啟。對於這個特定模型是否決定性地勝過對手，應持低至中等信心；「擊敗使用 20 倍資料的模型」是一個真實但幅度不大、僅限單一基準的結果。最值得記住的思想可信度很高：對於用程式碼繪圖的模型，邊畫邊看勝過閉眼作畫。

---

# 來源

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/