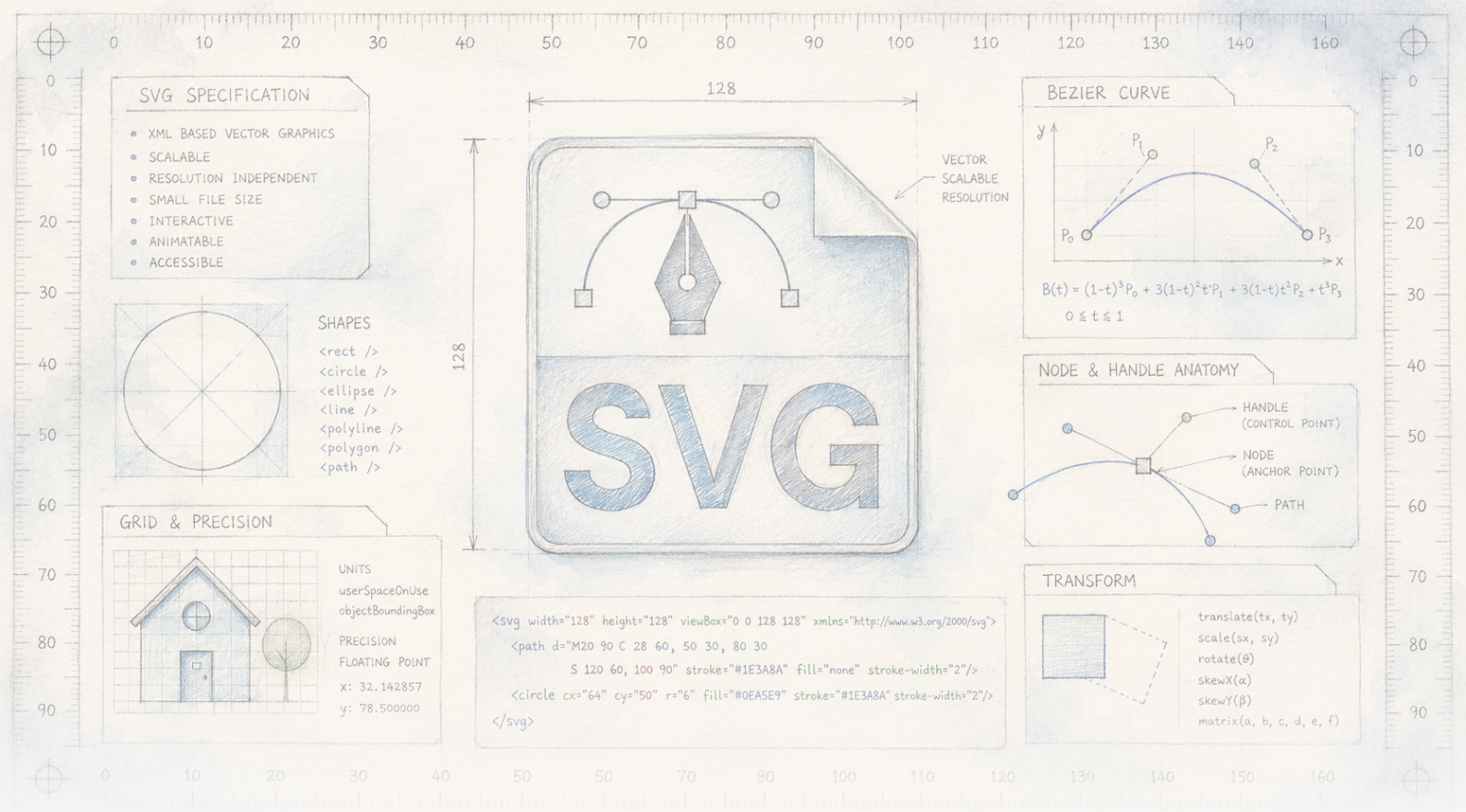


The Clean Paper

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

教繪圖 AI 看著畫布作畫

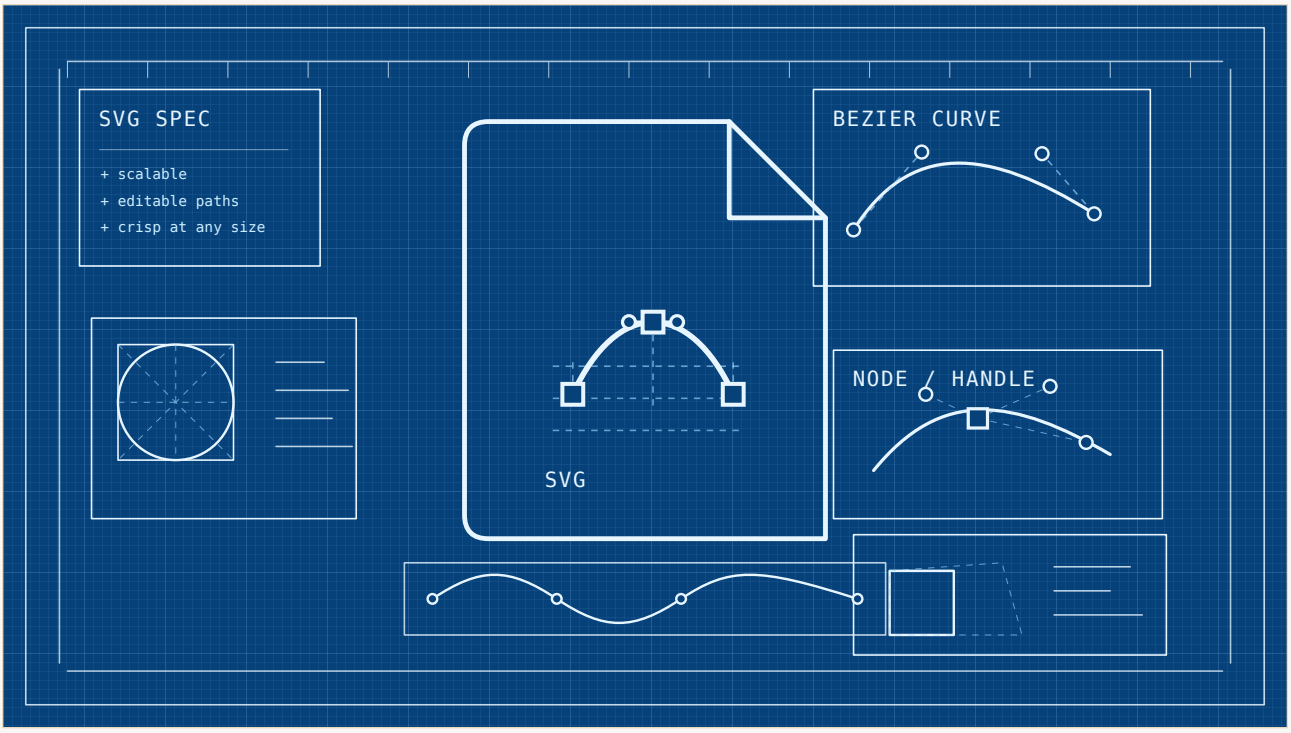
生成向量圖形的語言模型一直在盲畫——寫出繪圖命令，卻從未看見結果。一種新方法讓模型逐筆看著自己的畫布被填滿，而誠實的轉折是，簡單地賦予它眼睛反而讓結果變差：模型必須經過再訓練，學會怎樣使用視覺回饋。回報是一個能追平或略微勝過使用最多二十倍資料訓練的對手的模型——這是在單一基準上得到的真實、謹慎的結果，不是一場革命。

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

閉著眼睛畫畫

如果今天請某個 AI 模型替你畫一張圖，幕後可能發生兩種截然不同的事情。熟悉的影像生成器——那些能從一句話召喚出一張照片的模型——會直接繪製像素，而且能在作畫時看見畫布。但還有另一種比較安靜的畫法：模型根本不畫，而是寫下指令：這裡畫一個圓，那裡畫一條線，把這個形狀填成藍色。這些指令就是程式碼——也就是 Scalable Vector Graphics，或 SVG，網路上大多數圖示和標誌背後使用的格式——而它確實很有吸引力：結果不是固定的像素網格，而是一組可以無限縮放、重新上色和編輯，卻不會變模糊的形狀。



原始 SVG 藍圖作品：這張圖片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，負責寫出這些繪圖程式碼的模型都在盲畫。它一次寫出整串指令——圓、線、填色——卻從未渲染出來看看自己做了什麼。想像一下閉著眼睛描一張臉：你或許能把眼睛大致畫在該在的位置，卻無從察覺第二隻眼睛落在了臉頰上，也不知道你用來畫頭髮的形狀已經蓋住鼻子。這些模型的運作方式大致就是如此，因此它們經常產生完全有效、視覺上卻亂成一團的程式碼。

由 Guotao Liang 領導的團隊現在做了一件幾乎顯而易見、甚至有點滑稽的事：他們讓模型睜開眼睛。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，再把那張圖交還給模型，讓模型在畫下一筆之前先看見它。真正有意思的地方不在於這樣做有幫助，而在於他們必須做些什麼，才能讓它真的發揮作用。

> **如何評分一張圖？**
>
> 當一篇論文說它的模型「擊敗」了另一個模型時，合理的問題是：在哪方面擊敗？如何測量？

> 評判生成圖片確實很難——「畫一台筆電」沒有唯一正確的答案——所以這個領域依賴幾種自動評分方式，而每一種都只是人類觀看結果時所作判斷的不完美替代品。
> 這篇論文用了其中幾種。**FID** 會比較整批生成圖片與真實圖片的整體統計風格；分數越低越好，但它描述的是整批圖片，而不是其中任何一張。**CLIP score** 會請另一個 AI 判斷圖片是否符合提示詞——這很有用，但判斷力也取決於那個評審。**DINO**、**SSIM** 和 **LPIPS** 會把重建圖片與目標圖片比較，涵蓋從原始像素到學習得到的特徵等不同層次。
> 這些都不是真理。它們是代理指標，而且通常只會小幅變動。當你讀到一個模型得分 127.6、另一個是 128.8 時，這確實是朝預期方向的真實差異——但它只是小幅進步，而不是壓倒性勝利；而且這個數字與你的眼睛會給出的判斷只有鬆散的關聯。當標題寫著「擊敗以二十倍資料訓練的模型」時，這一點值得記住。

作者做了什麼

作者想修正的問題，就是這種盲畫。現有模型把撰寫 SVG 當成純文字任務：根據目前已有的程式碼預測下一段程式碼，卻從不渲染。這讓現代多模態模型本來就具備的強大「眼睛」——視覺編碼器——閒置不用。Render-in-the-Loop 將這項工作重組成逐步進行的視覺流程。每寫完一段繪圖程式碼，就把部分完成的 SVG 渲染成圖片，再以圖片形式餵回模型；於是模型會一邊實際查看目前的畫布，一邊選擇下一段程式碼。

他們的第一個發現帶有警示意味，而且值得肯定的是，他們坦白報告了這點：單純把這個迴圈接到現成模型上並不起作用。他們在沒有任何特殊訓練的情況下，把中間渲染結果餵給強大的通用模型，品質不但沒有改善，反而全面下降。一個從未被教導如何用眼睛完成這項工作的模型，不會突然就懂得使用自己的眼睛。

因此，大部分工作都在教學上。他們重新建構訓練資料，把每幅圖拆成許多小而有視覺意義的步驟——把複雜形狀拆成更簡單的部分，讓每個階段都有新的東西可看——接著用這些逐步序列微調一個八十億參數的開放模型（建構於 Qwen3-VL 之上）。他們稱之為 Visual Self-Feedback。作畫時，他們又加入第二個機制 Render-and-Verify：在接受每一筆新線條前，模型先渲染它，檢查它是否真的改變了圖片，還是只重複上一張。沒有改變畫面的線條會被丟棄；當再也沒有任何東西能帶來幫助時，模型就會收到停止的提示。值得注意的是，這一切都在相當小的資料集上運作——約 850,000 個樣本，不到一個競爭對手所用資料的一半，也只是另一個競爭對手所用資料的一小部分。

他們發現了什麼

- 以這種方式訓練後，模型畫得比它的盲畫版本更好——在失敗案例上尤其明顯：盲畫模型會把眼睛放在臉頰上，或跳過要求的長條圖，改而輸出一個通用螢幕。
- 在標準基準測試（MMSVGBench）中，這個方法的表現足以與強大的競爭方法抗衡，而且在多項指標上略勝一籌——包括 OmniSVG，它使用了超過兩倍的資料訓練；以及 InternSVG，它使用了約二十倍的資料訓練。
- 差距很小。例如在圖示集合上，它的主要圖片品質分數是 127.6，最佳競爭對手是 128.8；它的提

示詞匹配分數是 0.293，對手是 0.291——差異真實且一致，但幅度很小。
- 兩個新增部分都證明了自己的價值：關掉特殊訓練，或關掉作畫時的驗證，數字都會明顯下降。尤其是驗證步驟，能阻止模型陷入一遍又一遍重畫同一件事的困境。
- 作者自己強調的結果是效率——使用遠少於領先方法的訓練資料，卻能達到這樣的程度。

這不能證明什麼

- 它**並沒有**顯示讓模型「看見」是穩賺不賠的做法。事實上正好相反：論文自己的實驗顯示，沒有重新訓練的視覺回饋會讓結果變差。提升來自重新訓練，而不是單靠眼睛。
- 它**沒有**建立起大幅或決定性的領先。在大多數分數上，這個方法都與競爭對手難分高下；「擊敗以 20× 資料訓練的模型」這句話雖然正確，但只是單一基準測試中代理指標的小幅領先。
- 它**沒有**展示普遍的藝術能力。這是一個八十億參數的研究模型，在小幅固定解析度（224×224 像素）下繪製圖示和簡單插圖，而不是通用設計師。
- 與大型通用模型（GPT-5）的比較**不是同類比較**：那個模型並非為這項狹窄的程式碼繪圖任務打造或調校，因此在這裡勝過它，對任一模型的普遍能力都說明不了多少。
- 它**不是沒有代價**。每一步都渲染並重新讀取畫布，會使生成速度比一次盲目輸出全部程式碼更慢——作者也承認了這項成本。

證據有多強

- 在以自身條件進行的受控比較中，證據**很紮實**。消融實驗很乾淨：移除訓練，或移除驗證，數字就會下降——因此這兩項要素確實發揮了作者所宣稱的作用。
- **誠實面對自己的意外發現**。未經訓練就直接加入視覺回饋，反而會損害表現；這項發現有被報告，而不是被藏起來。它也是論文中最有意思的部分——有助於修正「輸入越多總是越好」的直覺。
- 在排行榜式的主張上，證據**偏薄弱**。勝過使用更多資料的競爭對手，其幅度很小，而且只出現在一個由其中一個競爭對手的作者所建立的基準測試上。單一測試集上的代理指標小幅差距，只能提供提示，不能算定論。
- **尚未在大規模或真實環境中測試**。這裡的一切都是低解析度的圖示和簡單插圖。相同想法是否適用於複雜、高解析度或真實世界的設計工作，仍留待未來研究——作者也明確如此表示。

為什麼重要

這篇論文的核心想法簡單得幾乎令人不好意思，而且遠遠不只適用於繪圖。如果一個程式要靠撰寫程式碼來生成某樣東西——網頁、圖表、圖解、3D 場景——它可以盲目寫完全部內容再祈求成功，也可以一邊渲染一邊修正方向。對人來說，閉合這個迴圈是再自然不過的事；我們會不斷瞥一眼頁面。對這些模型而言，這卻是相當近期才出現的做法。

這篇論文值得閱讀之處，在於它附帶的那個星號。給模型一種看見的方式，不等於教會它觀看。眼睛必須經過訓練，迴圈才會發揮作用——而且即使如此，回報也是真實但有限的：它成為更穩定的繪圖

者，而不是另一種藝術家。對任何正在打造「把描述轉成可編輯圖像」工具的人——也就是最終會用於應用程式圖示和插圖背後的那類工具——這個低調而實際的教訓才是有用的部分。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分更值得學習。

重點摘要

生成向量圖形的語言模型——也就是大多數網路圖示和標誌背後可編輯、可縮放的程式碼——過去通常都是「盲畫」：寫出所有繪圖指令，卻從未渲染它們來查看結果。由 Guotao Liang 領導的團隊提出 Render-in-the-Loop：每一步都渲染尚未完成的圖畫並餵回模型，讓模型一邊看著畫布，一邊畫出下一筆。他們核心且誠實的發現是，單純對現有模型做這件事反而會讓它變差；只有在重新訓練模型使用視覺回饋後，這項提升才會出現，而作畫時加入的檢查還會丟棄那些毫無改變的線條。重新訓練後的八十億參數模型，在標準基準測試上達到與競爭對手相當或略勝的表現，而那些對手使用了最多二十倍的資料——這是真實的結果，但在低解析度的圖示和簡單插圖上，代理分數的差距很小。作者強調、也值得記住的重點是效率：好好看見自己的作品，可以彌補單純堆疊資料規模的不足。

務實檢視

論文顯示了什麼：逐步重新訓練向量圖形模型，讓它渲染並查看自己尚未完成的圖畫，會比盲畫產生更好、更完整的結果——在一個標準基準測試上，以較少的訓練資料與使用更多資料的競爭對手抗衡，並略勝一籌。

合理但尚未證明的事：「查看自己的作品」廣泛而言是比擴大資料規模更好的方法；相同想法會對複雜、高解析度或真實世界的圖形有所幫助；基準測試中細小的差距確實反映出人類能察覺的差異。

它沒有顯示的事：視覺回饋單獨就有幫助（沒有重新訓練反而會造成傷害）；領先競爭對手的幅度很大或具有決定性；通用繪圖能力；與 GPT-5 這類並非為此任務打造的通用模型進行公平的正面對決。

主要限制：一個部分由競爭對手作者建立的基準測試；代理指標上的小幅差距；一個 8B 模型、圖示和 224×224 的簡單插圖；由於每一步都要渲染的迴圈而導致生成較慢。

一般讀者應該有多大信心？對於閉合這個迴圈——在繪畫時渲染並重新讀取——確實有幫助，而且必須把它訓練進模型，而不是只把它打開，讀者可以有高度信心。至於這個特定模型是否決定性地勝過競爭對手，信心應為低到中等；把「勝過 20× 資料」這句話視為一個真實但幅度溫和、來自單一基準測試的結果。對於唯一值得記住的想法，則可以有高度信心：對會繪圖的程式碼來說，一邊觀看勝過盲目作畫。

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/