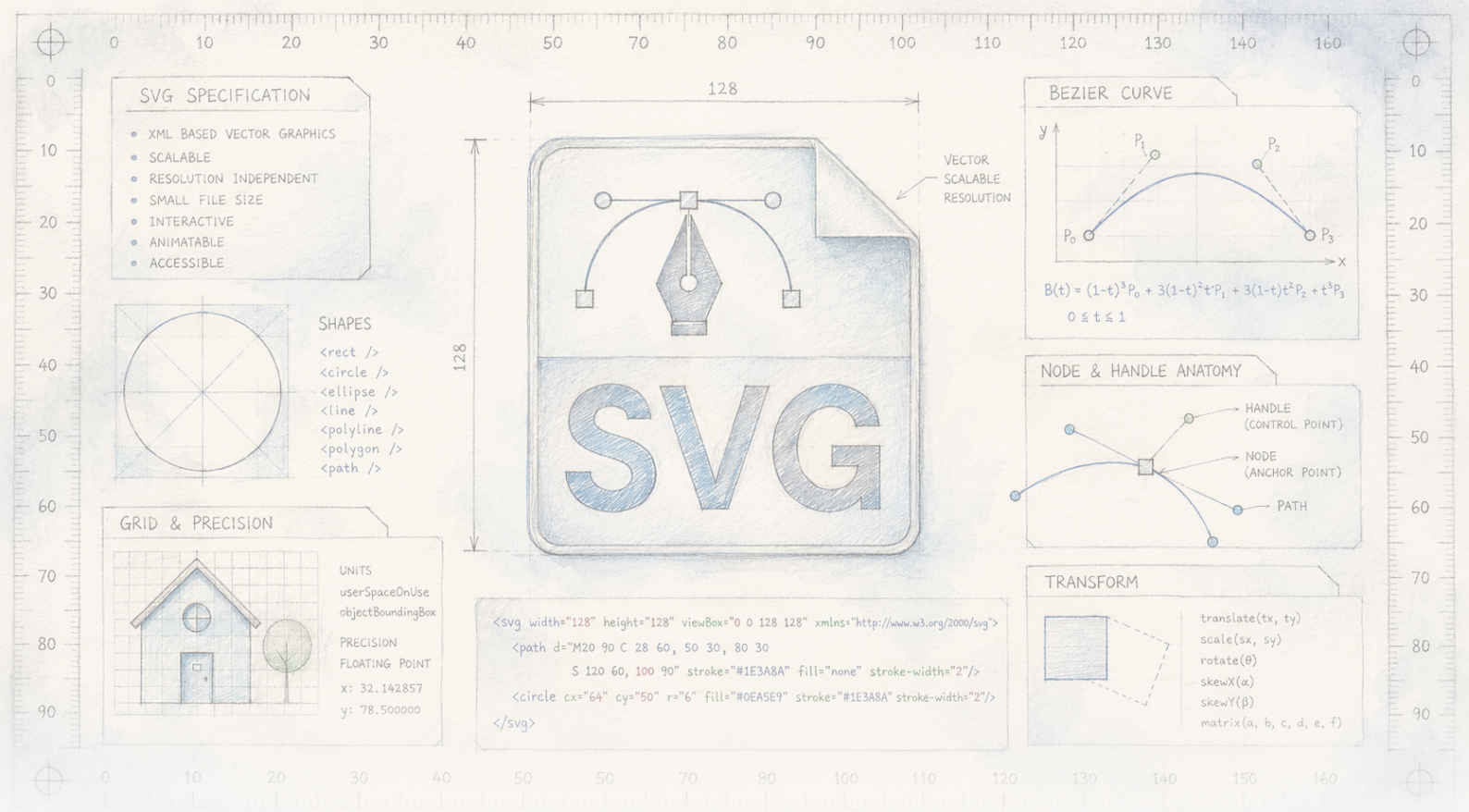


The Clean Paper

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

Навчити AI-художника дивитися на сторінку

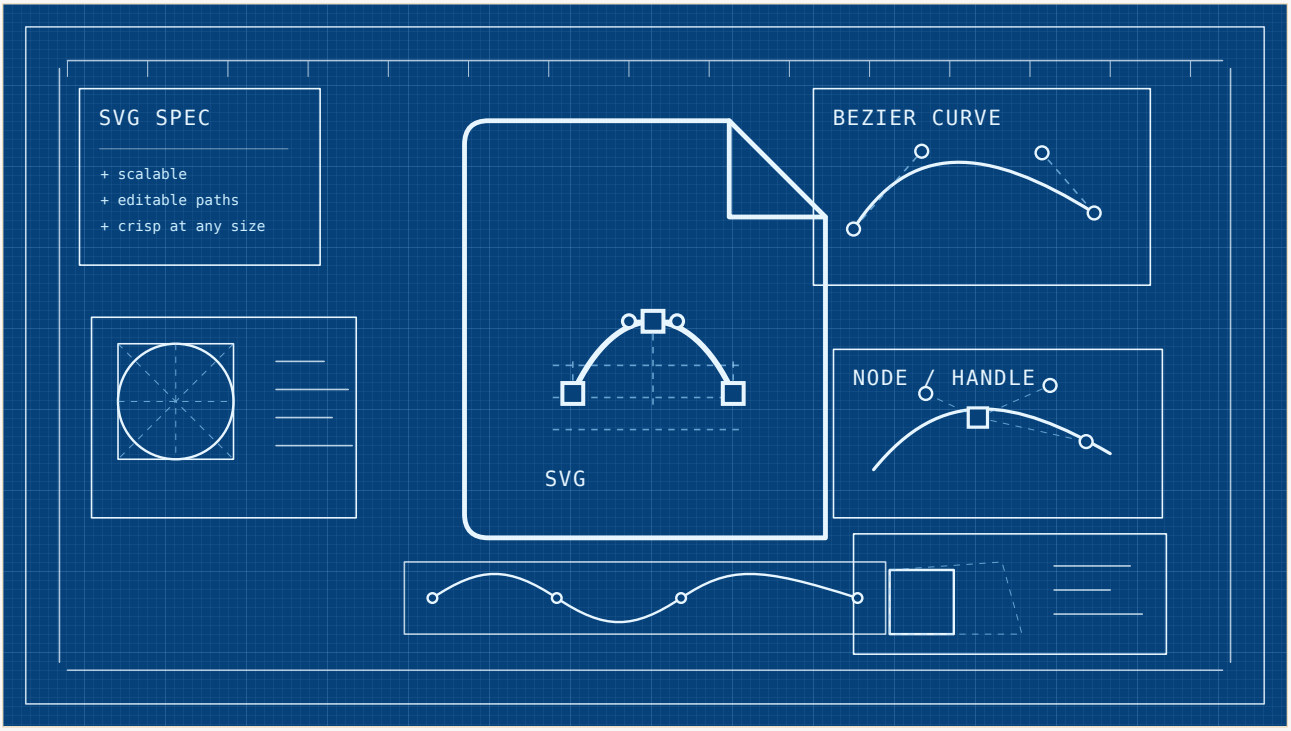
Мовні моделі, що генерують векторну графіку, роблять це наосліп — пишуть команди малювання, жодного разу не бачачи результату. Новий метод дозволяє моделі спостерігати, як її полотно заповнюється штрих за штрихом, і чесний поворот у тому, що просто дати їй очі робить результат гіршим: модель треба перенавчити ними користуватися. У підсумку вона дорівнює або трохи випереджає конкурентів, навчених на даних до двадцяти разів більшого обсягу, — реальний і обережний результат одного бенчмарку, а не революція.

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)

Малювати із заплющеними очима

Попросіть одну із сучасних AI-моделей намалювати вам картинку — і під капотом станеться одна з двох дуже різних речей. Знайомі генератори зображень, які створюють фотографію з речення, безпосередньо «малюють» пікселі й бачать полотно в процесі роботи. Але існує й другий, тихіший тип малювання, де модель узагалі нічого не малює. Вона пише інструкції: *намалюй тут коло, там лінію, заповни цю фігуру синім*. Ці інструкції — код, той самий Scalable Vector Graphics, або SVG, що лежить в основі більшості іконок і логотипів у вебі. Його перевага реальна: результат — не фіксована сітка пікселів, а набір фігур, які можна безкінечно масштабувати, перефарбовувати й редагувати без розмиття.



Оригінальна SVG-ілюстрація у стилі технічного креслення: саме зображення є редагованим векторним файлом, побудованим із контурів, ліній сітки, вузлів і керувальних ручок, а не з фіксованих пікселів.

Laura Nesso / The Clean Paper Permission on file

Проблема в тому, що донедавна модель, яка писала такий код для малюнка, робила це наосліп. Вона видавала всю послідовність команд за один прохід — коло, лінія, заливка — і жодного разу не рендерила результат, щоб подивитися, що вийшло. Уявіть, що малюєте обличчя із заплющеними очима: ви, можливо, приблизно поставите очі там, де вони мають бути, але не помітите, що друге опинилося на щоці або що фігура для волосся тепер перекриває ніс. Приблизно так ці моделі й працювали, тому вони часто створювали цілком коректний код, який візуально був безладом.

Команда під керівництвом Guotao Liang зробила майже комічно очевидну річ: дозволила моделі розплющити очі. Їхній метод *Render-in-the-Loop* після кожного кроку рендерить

незавершений малюнок і повертає цю картинку моделі перед наступним штрихом. По-справжньому цікаво не те, що це допомагає, — а те, що їм довелося зробити, аби це взагалі почало допомагати.

Як оцінюють малюнок?

Коли стаття каже, що її модель «перемагає» іншу, справедливо запитати: перемагає в чому й за яким вимірюванням? Оцінювати згенероване зображення справді складно — немає єдиної правильної відповіді на прохання «намалюй ноутбук», — тому галузь спирається на кілька автоматичних метрик, кожна з яких лише недосконало замінює людину, що дивиться на результат.

У цій роботі з'являються кілька таких метрик. **FID** порівнює загальні статистичні властивості цілої партії згенерованих зображень із реальними; менше — краще, але це оцінка партії, не окремої картинки. **CLIP score** запитує іншу AI-модель, наскільки зображення відповідає текстовому запиту — корисно, але лише настільки, наскільки добрий цей суддя. **DINO**, **SSIM** і **LPIPS** порівнюють реконструйоване зображення з цільовим на рівнях від сирих пікселів до навчених ознак.

Жодна з цих метрик не є істиною. Це проксі-показники, і вони часто змінюються малими кроками. Коли ви читаєте, що одна модель отримала 127,6, а інша 128,8, це реальна різниця в потрібному напрямку — але невеликий поштовх, а не обвал, до того ж у числі, яке лише приблизно відбиває те, що сказало б ваше око. Це варто пам'ятати, коли заголовок звучить як «перемагає модель, навчену на двадцятикратно більшій кількості даних».

Що зробили автори

Проблема, яку автори хотіли виправити, — саме це сліпе малювання. Попередні моделі трактували написання SVG як чисто текстове завдання: передбачити наступний шматок коду з уже написаного й ніколи його не рендерити. Через це без роботи залишалися потужні «очі» — візуальний енкодер, який уже мають сучасні мультимодальні моделі. Render-in-the-Loop перебудовує задачу на покроковий візуальний процес. Після кожного фрагмента коду частковий SVG рендериться як зображення й подається назад моделі, тож наступний фрагмент вона обирає, реально дивлячись на поточне полотно.

Їхній перший результат — застереження, і, на їхню честь, вони повідомляють його прямо: просто прикрутити цей цикл до готової моделі не працює. Коли дослідники подавали проміжні рендери сильним універсальним моделям без спеціального навчання, якість не поліпшилася — вона *погіршилася* за всіма показниками. Модель, яку ніколи не вчили використовувати очі саме для цього завдання, не починає раптом уміти це робити.

Тому більша частина роботи — у навчанні. Автори перебудовують тренувальні дані

так, щоб кожен малюнок був розбитий на багато малих, візуально осмислених кроків — складні фігури ділять на простіші частини, щоб на кожній стадії було щось нове для перегляду, — а потім донавчають відкриту модель із вісьмома мільярдами параметрів (на основі Qwen3-VL) на цих покрокових послідовностях. Це вони називають Visual Self-Feedback. Додають і другий механізм під час малювання — Render-and-Verify: перш ніж прийняти кожен новий штрих, модель рендерить його й перевіряє, чи справді він змінив картинку, чи лише повторив попередній. Штрихи, що нічого не додають, відкидаються, а коли подальші кроки вже не допомагають, моделі пропонують зупинитися. Примітно, що все це працює на доволі невеликому наборі — приблизно 850 000 прикладів, менше половини того, що використовував один конкурент, і мала частка даних іншого.

Що вони виявили

- Навчена таким способом модель малює краще за свій «сліпий» варіант — особливо помітно на типових помилках, коли сліпа модель ставить око на щоку або ігнорує запитаний стовпчиковий графік і видає натомість звичайний монітор.
- На стандартному бенчмарку MMSVGBench метод конкурентоспроможний і за кількома метриками трохи випереджає сильних суперників — зокрема OmniSVG, навчений на більш ніж удвічі більшому наборі даних, і InternSVG, навчений приблизно на двадцятикратно більшому.
- Відрив невеликий. На наборі іконок, наприклад, головна метрика якості зображення становить 127,6 проти 128,8 у найкращого конкурента, а показник відповідності запиту — 0,293 проти 0,291: реальні й послідовні відмінності, але тонкі.
- Обидва додані компоненти виправдовують своє місце: вимкніть спеціальне навчання або перевірку під час малювання — і показники помітно падають. Перевірка особливо добре не дає моделі застрягнути, знову й знову перемальовуючи те саме.
- Результат, який найбільше підкреслюють самі автори, — ефективність: такого рівня досягнуто з набагато меншою кількістю навчальних даних, ніж використовували лідери.

Чого це не доводить

- Це **не показує**, що дати моделі «зір» — безкоштовна перемога. Навпаки: власний експеримент статті показує, що візуальний зворотний зв’язок *без* перенавчання погіршує результат. Виграш походить від навчання, а не від самих очей.
- Це **не встановлює великої чи вирішальної переваги**. За більшістю метрик метод іде нога в ногу з конкурентами; «перемагає модель із 20× більшими даними» — правда, але йдеться про невеликі зрушення проксі-метрик на одному бенчмарку.
- Це **не демонструє загальної художньої здатності**. Це дослідницька модель

на вісім мільярдів параметрів, що малює іконки й прості ілюстрації з малою фіксованою роздільною здатністю 224×224 пікселі, а не універсальний дизайнер.
- Порівняння з великою універсальною моделлю GPT-5 **не є прямим порівнянням однакових речей**: GPT-5 не створений і не налаштований під це вузьке завдання малювання кодом, тож перемога тут мало говорить про будь-яку з моделей загалом.
- Це **не безкоштовно з погляду обчислень**. Рендерити й заново читати полотно на кожному кроці повільніше, ніж один раз наосліп видати весь код, — і автори це визнають.

Наскільки сильні докази

- **Міцні** там, де є контрольоване порівняння на власних умовах. Абляції чисті: заберіть навчання або перевірку — показники падають, тож обидві складові справді виконують заявлену роботу.
- **Чесні щодо власного сюрпризу.** Результат, що наївний візуальний зворотний зв'язок *шкодить*, не захований, а прямо повідомлений; це, мабуть, найцікавіша частина статті й корисне виправлення інтуїції, ніби більше вхідної інформації завжди краще.
- **Тонкі як твердження про місце в рейтингу.** Перемоги над конкурентами з більшими наборами даних невеликі й отримані на одному бенчмарку, створеному авторами однієї з конкуруючих систем. Малі відриви в проксі-метриках на одному тестовому наборі — це натяк, а не остаточний вирок.
- **Не перевірені на великому масштабі й у реальній роботі.** Тут лише іконки й прості ілюстрації з низькою роздільною здатністю. Чи працюватиме та сама ідея для складного, високороздільного або реального дизайнерського завдання, лишається майбутньою роботою — автори самі це кажуть.

Чому це важливо

Ідея в центрі цієї статті майже сороміцьки проста й виходить далеко за межі малювання. Якщо програма генерує щось, пишучи код — вебсторінку, графік, діаграму, 3D-сцену, — вона може або написати все наосліп і сподіватися, або рендерити в процесі й коригувати курс. Для людини замкнути цей цикл природно: ми постійно поглядаємо на сторінку. Для таких моделей це напрочуд недавній хід.

Цінність статті — у зірочці біля цієї ідеї. Дати моделі можливість бачити — не те саме, що навчити її дивитися. «Очі» треба тренувати, і лише тоді цикл окупається — та навіть тоді виграш реальний, але помірний: стабільніший кресляр, а не інший вид художника. Для тих, хто будує інструменти, що перетворюють опис на редаговану графіку — саме таку, яка потім стає іконками й ілюстраціями застосунків, — корисний саме тихий практичний урок. Виграш тут прийшов від дешевшого й розумнішого

навчання, а не від більшої кількості даних. Це цінніший урок, ніж ще один бал у таблиці лідерів.

Короткий підсумок

Мовні моделі, що генерують векторну графіку — редагований, масштабований код за більшістю вебіконок і логотипів, — традиційно робили це «наосліп»: писали всі команди малювання, жодного разу не рендерячи результат. Команда під керівництвом Guotao Liang пропонує Render-in-the-Loop: після кожного кроку рендерити незавершений малюнок і подавати його назад, щоб наступний штрих модель робила, дивлячись на полотно. Їхній головний і чесний результат: просто додати це до наявної моделі робить її *гіршою*; виграш з'являється лише після перенавчання на використання візуального зворотного зв'язку, разом із перевіркою під час малювання, яка відкидає штрихи, що нічого не змінюють. Перенавчена модель на вісім мільярдів параметрів на стандартному бенчмарку дорівнює або трохи випереджає конкурентів, навчених на даних до двадцяти разів більшого обсягу — справжній результат, але з малими відривами в проксі-метриках, для іконок і простих ілюстрацій із низькою роздільною здатністю. Висновок, який підкреслюють автори й який варто зберегти, — про ефективність: уміння добре бачити власну роботу частково замінює просте нарощування масштабу даних.

Тверезa оцінка

Що показує стаття: Перенавчання моделі векторної графіки так, щоб вона крок за кроком рендерила й розглядала власний незавершений малюнок, дає кращі й повніші результати, ніж сліпе малювання, — конкурентні й трохи кращі за суперників із більшими наборами даних на одному стандартному бенчмарку, при меншій кількості тренувальних даних.

Що правдоподібне, але не доведене: Що «дивитися на власну роботу» загалом краща стратегія, ніж просто масштабувати дані; що та сама ідея допоможе зі складною, високороздільною чи реальною графікою; що малі відмінності в бенчмарку відповідають різниці, яку людина реально помітить.

Чого робота не показує: Що візуальний зворотний зв'язок допомагає сам по собі (без перенавчання він шкодить); що перевага над конкурентами велика або вирішальна; загального вміння малювати; справедливого прямого порівняння з універсальними моделями на кшталт GPT-5, які не створені для цього завдання.

Основні обмеження: Один бенчмарк, частково створений авторами конкуруючої системи; малі відриви в проксі-метриках; модель 8B, іконки та прості ілюстрації 224×224; повільніше генерування через цикл рендерингу на кожному кроці.

Скільки довіри варто мати звичайному читачеві? Високу, що замикання циклу — рендерити й перечитувати результат у процесі — справді допомагає і що цього треба навчати, а не просто ввімкнути. Низьку або помірну щодо того, що саме ця модель вирішально перемагає конкурентів; фразу «перемагає при 20× меншій кількості даних» слід сприймати як реальний, але скромний результат одного бенчмарку. Високу щодо однієї ідеї, яку варто пам’ятати: для коду, що малює, дивитися в процесі краще, ніж малювати наосліп.

Джерела

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/