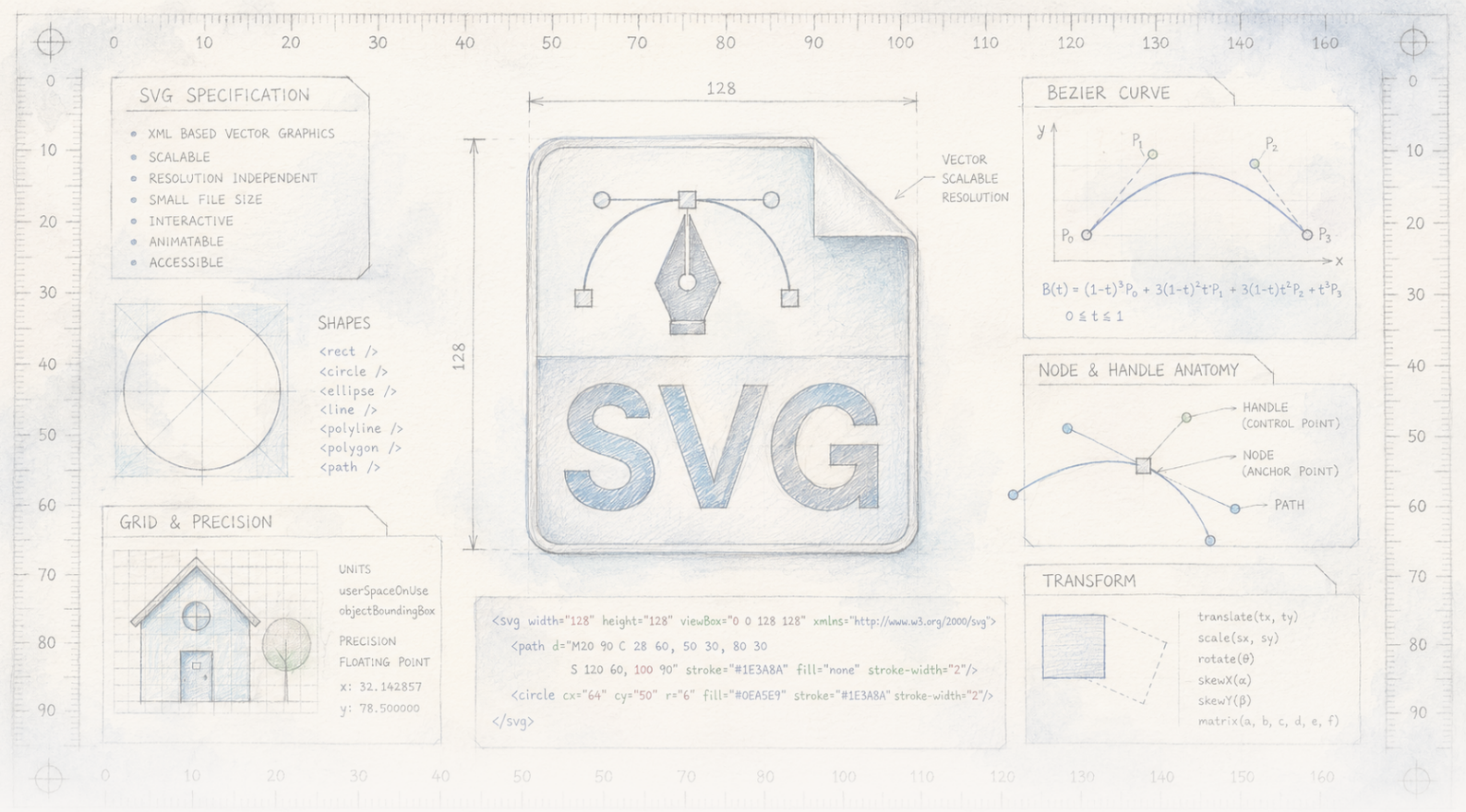


The Clean Paper

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

Megtanítani egy rajzoló MI-t arra, hogy nézze a lapot

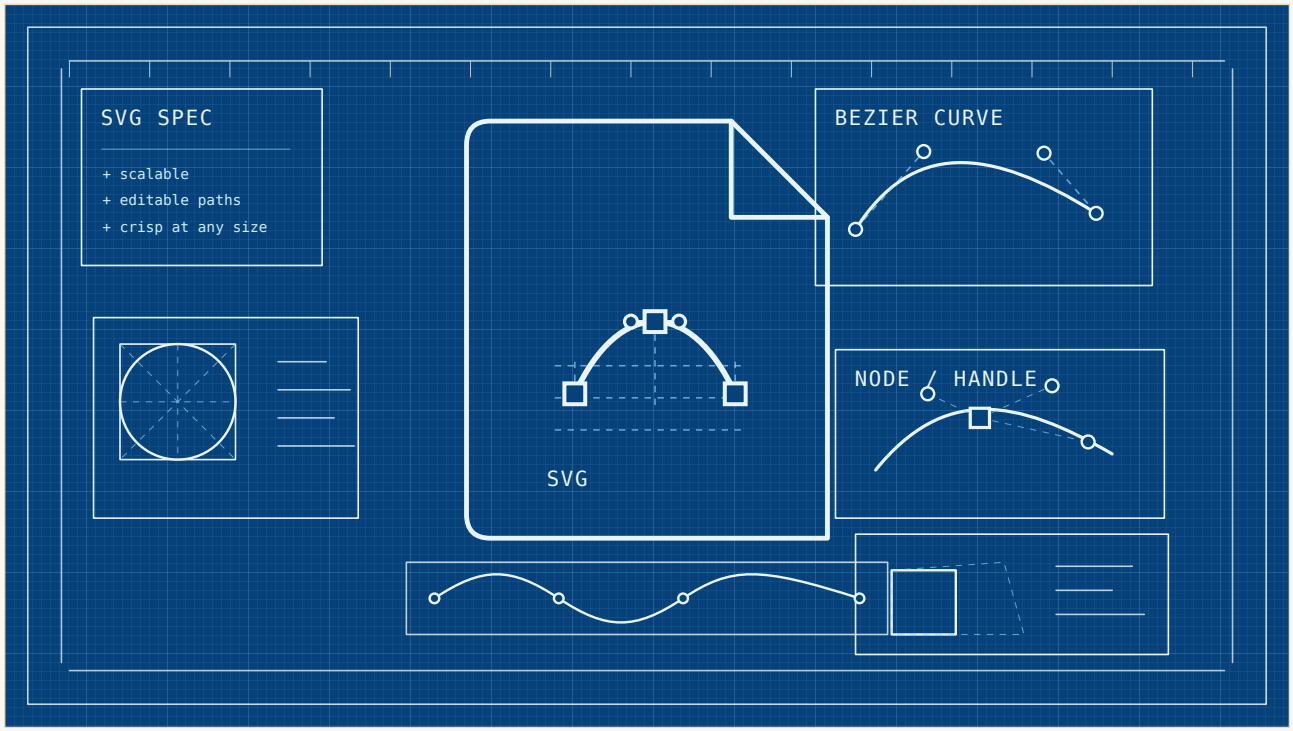
A vektorgrafikát generáló nyelvi modellek vakon dolgoznak: úgy írják ki a rajzolási parancsokat, hogy közben nem látják az eredményt. Egy új módszer lehetővé teszi, hogy a modell vonásról vonásra figyelje, hogyan telik meg a vászon. Az őszinte csavar az, hogy pusztán attól, hogy szemet kap, még rosszabb lesz: újra kell tanítani arra, hogyan használja a látványt. A jutalom egy olyan modell, amely egy benchmarkon utoléri vagy kissé megelőzi az akár hússzor több adattal tanított riválisokat — valódi, gondosan körülhatárolt eredmény, nem forradalom.

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)

Rajzolás csukott szemmel

Ha megkérjük valamelyik mai MI-modellt, hogy rajzoljon nekünk egy képet, a háttérben két egészen eltérő dolog történhet. A jól ismert képgenerátorok — amelyek egy mondatból fényképszerű képet varázsolnak — közvetlenül pixeleket festenek, és munka közben „látják” a vásznat. Létezik azonban a rajzolásnak egy másik, csendesebb fajtája is, ahol a modell egyáltalán nem fest. Utasításokat ír: *rajzolj ide egy kört, oda egy vonalat, ezt az alakzatot töltsd ki kékkel.* Ezek az utasítások kódok — ugyanaz a Scalable Vector Graphics, vagyis SVG, amely a weben található ikonok és logók nagy része mögött áll. Ennek valódi előnye van: az eredmény nem rögzített pixelrács, hanem olyan alakzatok együttese, amelyeket tetszőlegesen átméretezhetünk, átszínezhetünk és szerkeszthetünk anélkül, hogy elmosódnának.



Eredeti SVG-tervrajz: maga a kép is szerkeszthető vektorfájl, amely rögzített pixelek helyett útvonalakból, rácsvonalakból, csomópontokból és fogópontokból épül fel.

Laura Nesso / The Clean Paper Permission on file

A bökkenő az, hogy egészen a közelmúltig az ilyen rajzkódot író modell vakon dolgozott. Egyetlen menetben kibocsátotta az utasítások teljes sorozatát — kör, vonal, kitöltés — anélkül, hogy közben renderelte volna, amit készített. Olyan ez, mintha csukott szemmel próbálnánk arcot rajzolni: nagyjából oda kerülhetnek a szemek, ahová kell, de nem vehetjük észre, ha a második a pofára csúszott, vagy ha a hajnak szánt forma már az orrot takarja. Lényegében így működtek ezek a modellek, ezért fordult elő olyan gyakran, hogy tökéletesen érvényes kódot állítottak elő, amely vizuálisan mégis kusza lett.

Guotao Liang vezetésével egy kutatócsoport most megtette a szinte komikusan kézenfekvő lépést: kinyitották a modell szemét. *Render-in-the-Loop* módszerük minden lépés után rendereli a félkész

rajzot, majd ezt a képet visszaadja a modellnek, mielőtt az megrajzolná a következő vonást. Az igazán érdekes rész nem az, hogy ez segít — hanem az, hogy mit kellett tenniük ahhoz, hogy egyáltalán segítsen.

Hogyan pontozunk egy rajzot?

Amikor egy tanulmány azt állítja, hogy a modellje „jobb” egy másiknál, joggal kérdezhetjük: miben jobb, és hogyan mérték? Egy generált kép megítélése valóban nehéz — nincs egyetlen helyes válasz arra, hogy „rajzolj egy laptopot” — ezért a terület néhány automatikus mutatóra támaszkodik, amelyek mind tökéletlen helyettesítői annak, amit egy ember látna.
A tanulmányban több ilyen is szerepel. A **FID** a generált képek egész csoportjának statisztikai jellegét hasonlítja a valódi képekéhez; az alacsonyabb érték jobb, de ez a teljes halmazt írja le, nem egyetlen képet. A **CLIP score** egy külön MI-t kérdez meg arról, hogy egy kép mennyire illik a szöveges prompthoz — hasznos, de csak annyira éles mérce, amennyire maga a bírómodell. A **DINO**, az **SSIM** és az **LPIPS** egy rekonstruált képet hasonlít egy célképhez, a nyers pixelektől a tanult jellemzőkig különböző szinteken.
Ezek egyike sem maga az igazság. Helyettesítő mérőszámok, és gyakran csak kis lépésekben változnak. Ha azt olvassuk, hogy az egyik modell 127,6 pontot ér el, míg a másik 128,8-at, az valódi különbség a kívánt irányban — de finom elmozdulás, nem földcsuszamlás, ráadásul egy olyan számban, amely csak lazán követi azt, amit a szemünk mondana. Ezt érdemes észben tartani, amikor a főcím az, hogy „felülmúl egy húszszor több adattal betanított modellt”.

Mit csináltak a szerzők

A szerzők a vak rajzolás problémáját akarták kijavítani. A meglévő modellek az SVG írását tisztán szöveges feladatként kezelik: a már elkészült kód alapján megjósolják a következő kódrészletet, és közben soha nem renderelik a képet. Így kihasználatlanul maradnak a modern multimodális modellekben már meglévő erős „szemek”, vagyis a vizuális enkóder. A Render-in-the-Loop lépésről lépésre végzett vizuális folyamattá alakítja a feladatot. Minden rajzkódrészlet után a részleges SVG képpé renderelődik, és visszakerül a modellhez, így a következő részletet úgy választja ki, hogy ténylegesen látja az addigi vásznat.

Első eredményük óvatosságra int, és érdemükre legyen mondva, világosan közlik: nem működik, ha ezt a hurkot egyszerűen ráillesztik egy kész, általános modellre. Amikor speciális betanítás nélkül adtak köztes renderelt képeket erős általános modelleknek, a minőség nem javult — mindenhol *rom-lott*. Egy modell, amelyet soha nem tanítottak meg arra, hogy ebben a feladatban használja a szemét, nem tudja ezt hirtelen magától.

A munka nagy része ezért maga a tanítás. Úgy építik át a tanítóadatokat, hogy minden rajz sok apró, vizuálisan értelmes lépésre bomoljon — a bonyolult alakzatokat egyszerűbb részekre szedik, hogy minden fázisban legyen valami új, amit látni lehet —, majd egy nyolcmilliárd paraméteres nyílt modellt (Qwen3-VL alapokon) finomhangolnak ezeken a lépésenkénti sorozatokon. Ezt Visual Self-Feedbacknek nevezik. Ehhez rajzolás közben egy második mechanizmust, a Render-and-Verifyt

is hozzáadják: mielőtt elfogadnák az új vonást, a modell rendereli, és ellenőrzi, hogy valóban megváltoztatta-e a képet, vagy csak megismételte az előzőt. A semmit hozzá nem adó vonásokat eldobják, és amikor már semmi további nem javít a képen, a modellt leállásra késztetik. Figyelemre méltó, hogy mindez viszonylag kis adathalmazon fut: körülbelül 850 000 példán, ami kevesebb mint fele az egyik riválisadatainak, és csak töredéke egy másikénak.

Mit találtak

- Az így betanított modell jobban rajzol, mint vak megfelelője — ez leginkább a hibás esetekben látszik, amikor a vak modell például egy szemet a pofára tesz, vagy kihagy egy kért oszlopdiagramot, és helyette általános monitort ad.
- A szokásos benchmarkon (MMSVGBench) a módszer versenyképes az erős riválisokkal, és több mérőszám szerint kissé előttük is végez — köztük az OmniSVG előtt, amelyet több mint kétszer annyi adattal, illetve az InternSVG előtt, amelyet nagyjából húszszor annyi adattal tanítottak.
- A különbségek kicsik. Az ikonkészleten például a fő képminőségi pontszáma 127,6, míg a legjobb riválisé 128,8; a prompthoz való illeszkedés pontszáma 0,293 a 0,291-gyel szemben — valódi és következetes előny, de keskeny.
- Mindkét hozzáadott elemnek van szerepe: ha kikapcsolják a speciális betanítást vagy a rajzolás közbeni ellenőrzést, a számok mérhetően romlanak. Az ellenőrzési lépés különösen abban segít, hogy a modell ne ragadjon bele ugyanannak az elemnek az újra és újra megrajzolásába.
- A szerzők maguk az adathatékonyságot hangsúlyozzák: ennyire jutottak jóval kevesebb tanítóadattal, mint amennyit az élmezőny modelljei használtak.

Mit nem bizonyít ez

- **Nem** mutatja, hogy önmagában nyereség, ha egy modell „lát”. Épp ellenkezőleg: a tanulmány saját kísérlete szerint az újratanítás nélküli vizuális visszacsatolás ront a teljesítményen. A javulást a betanítás adja, nem pusztán a szemek.
- **Nem** igazol nagy vagy döntő előnyt. A legtöbb pontszámban a módszer fej fej mellett halad a riválisokkal; igaz, hogy „felülmúl egy 20× több adattal tanított modellt”, de csak kis elmozdulásokkal, helyettesítő mérőszámokon és egyetlen benchmarkon.
- **Nem** bizonyít általános művészi képességet. Ez egy nyolcmilliárd paraméteres kutatási modell, amely ikonokat és egyszerű illusztrációkat rajzol kis, rögzített felbontáson (224×224 pixel), nem általános célú tervező.
- A nagy általános modellel (GPT-5) való összehasonlítás **nem** teljesen azonos feltételek mellett történik: azt a modellt nem erre a szűk kódrajzolási feladatra tervezték vagy hangolták, ezért az itteni veresége egyik modell általános képességeiről sem mond sokat.
- **Nem** ingyenes a javulás. A vászon minden lépésnél történő renderelése és újraolvasása lassabbá teszi a generálást, mint amikor a modell vakon, egyetlen menetben írja ki a kódot — ezt a költséget a szerzők is elismerik.

Mennyire erős a bizonyíték

- **Szilárd** ott, ahol a saját feltételei között kontrollált összehasonlításról van szó. Az ablatív vizsgálatok tiszták: ha eltávolítják a betanítást vagy az ellenőrzést, a számok romlanak — vagyis valóban ez a két összetevő végzi azt a munkát, amelyet a szerzők nekik tulajdonítanak.
- **Őszintén kezeli a meglepetést.** Nem rejtik el azt az eredményt, hogy a naiv vizuális visszacsatolás *árt*. Ez a tanulmány legérdekesebb része, és hasznos korrekciója annak az ösztönös feltételezésnek, hogy a több bemenet mindig jobb.
- **Vékonyabb**, amikor ranglista-elsőségről van szó. A nagyobb adathalmazon tanított riválisok feletti győzelmek kicsik, és egyetlen benchmarkon jelennek meg, amelyet ráadásul az egyik rivális modell szerzői készítettek. Kis különbségek helyettesítő mérőszámokon, egyetlen tesztkészleten: érdekes jelzés, nem lezárt kérdés.
- **Nincs nagy léptékben és valós használatban tesztelve.** Itt minden ikonokra és egyszerű, kis felbontású illusztrációkra vonatkozik. Hogy ugyanez az ötlet működik-e összetett, nagy felbontású vagy valós tervezési feladatokban, jövőbeli kutatás kérdése — ezt a szerzők is kimondják.

Miért fontos

A tanulmány központi ötlete szinte zavarba ejtően egyszerű, és jóval túlmutat a rajzoláson. Ha egy program kód írásával hoz létre valamit — weboldalt, diagramot, ábrát, 3D-jelenetet —, akkor vagy vakon megírhat mindent és remélheti a legjobbat, vagy menet közben renderelheti az eredményt, és korrigálhatja az irányt. Egy ember számára ennek a huroknak a bezárása magától értetődő: folyamatosan rápillantunk a lapra. Ezeknél a modelleknél viszont meglepően új lépés.

A tanulmányt az a csillaggal jelzett kiegészítés teszi igazán érdekessé, amelyet ehhez hozzáfűz. Az, hogy egy modellnek lehetőséget adunk a látásra, nem ugyanaz, mint megtanítani nézni. A „szemeket” be kell tanítani, és csak ezután hoz hasznot a visszacsatolási hurok — és akkor is valós, de mértéktartó hasznot: megbízhatóbb rajzolót, nem egészen másféle művészt. Azok számára, akik leírásból szerkeszthető grafikát készítő eszközöket építenek — például alkalmazások ikonjaihoz és illusztrációihoz —, éppen a csendes, gyakorlati tanulság a hasznos. A javulás itt olcsóbb, okosabb betanításból jött, nem több adatból. Ez többet ér, mint még egy pont a ranglistán.

Röviden

A vektorgrafikát — vagyis a legtöbb webes ikon és logó mögötti szerkeszthető, átméretezhető kódot — generáló nyelvi modellek hagyományosan „vakon” dolgoztak: kiírták az összes rajzolási utasítást anélkül, hogy valaha renderelték volna az eredményt. Guotao Liang és munkatársai a Render-in-the-Loop módszert javasolják: minden lépés után renderelik a félkész rajzot, majd visszaadják a modellnek, hogy a következő vonást már a vásznat nézve készítse el. Központi és őszinte eredményük az, hogy ha ezt egyszerűen rákapcsolják egy meglévő modellre, az *rosszabb* lesz; javulás csak akkor

jelenik meg, ha a modellt újratanítják a vizuális visszacsatolás használatára, és ezt egy olyan rajzolás közbeni ellenőrzés segíti, amely eldobja a semmit sem változtató vonásokat. Az újratanított, nyolcmilliárd paraméteres modell egy szokásos benchmarkon utoléri vagy kissé felülmúlja a akár húszszor több adattal tanított riválisokat — valódi eredmény, de kis különbségekkel, helyettesítő pontszámokon, kis felbontású ikonok és egyszerű illusztrációk esetében. A szerzők által hangsúlyozott, megőrzésre érdemes tanulság az adathatékonyságról szól: ha egy modell jól tudja nézni a saját munkáját, az részben kiválthatja a puszta adatmennyiséget.

Tárgyilagos mérleg

Mit mutat a tanulmány: Ha egy vektorgrafikai modellt újratanítanak arra, hogy lépésről lépésre renderelje és megnézze a saját félkész rajzát, jobb és teljesebb eredményt ad, mint vak rajzoláskor — egy szokásos benchmarkon kevesebb tanítóadatból versenyképes a nagyobb adathalmazokon tanított riválisokkal, és kissé meg is előzi őket.

Mi hihető, de nincs bizonyítva: Hogy a „nézd a saját munkádat” általában jobb recept, mint egyszerűen több adatot használni; hogy ugyanez az ötlet összetett, nagy felbontású vagy valós grafikai feladatokon is segít; hogy a kis benchmark-különbséget egy ember ténylegesen észrevenné.

Mit nem mutat: Hogy a vizuális visszacsatolás önmagában segít (újratanítás nélkül ront); hogy a riválisok feletti előny nagy vagy döntő; általános célú rajzkészséget; tisztességes, azonos feltételek melletti összehasonlítást olyan általános modellekkel, mint a GPT-5, amelyeket nem erre a feladatra építettek.

Fő korlátok: Egy benchmark, amelyet részben egy rivális szerzői készítettek; kis különbségek helyettesítő mérőszámokon; egy 8B modell; ikonok és egyszerű illusztrációk 224×224 pixelen; lassabb generálás a minden lépésben végzett renderelés miatt.

Mennyire bízhat benne egy általános olvasó? Nagy bizalommal abban, hogy a hurok bezárása — vagyis a menet közbeni renderelés és újranézés — valóban segít, és ezt be kell tanítani, nem elég egyszerűen bekapcsolni. Alacsony-közepes bizalommal abban, hogy ez a konkrét modell döntően jobb a riválisainál; a „20× több adatot ver” állítást valódi, de szerény, egyetlen benchmarkon mért eredményként érdemes kezelni. És nagy bizalommal abban az egyetlen gondolatban, amelyet érdemes megjegyezni: rajzoló kód esetén jobb menet közben nézni, mint vakon rajzolni.

Források

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/