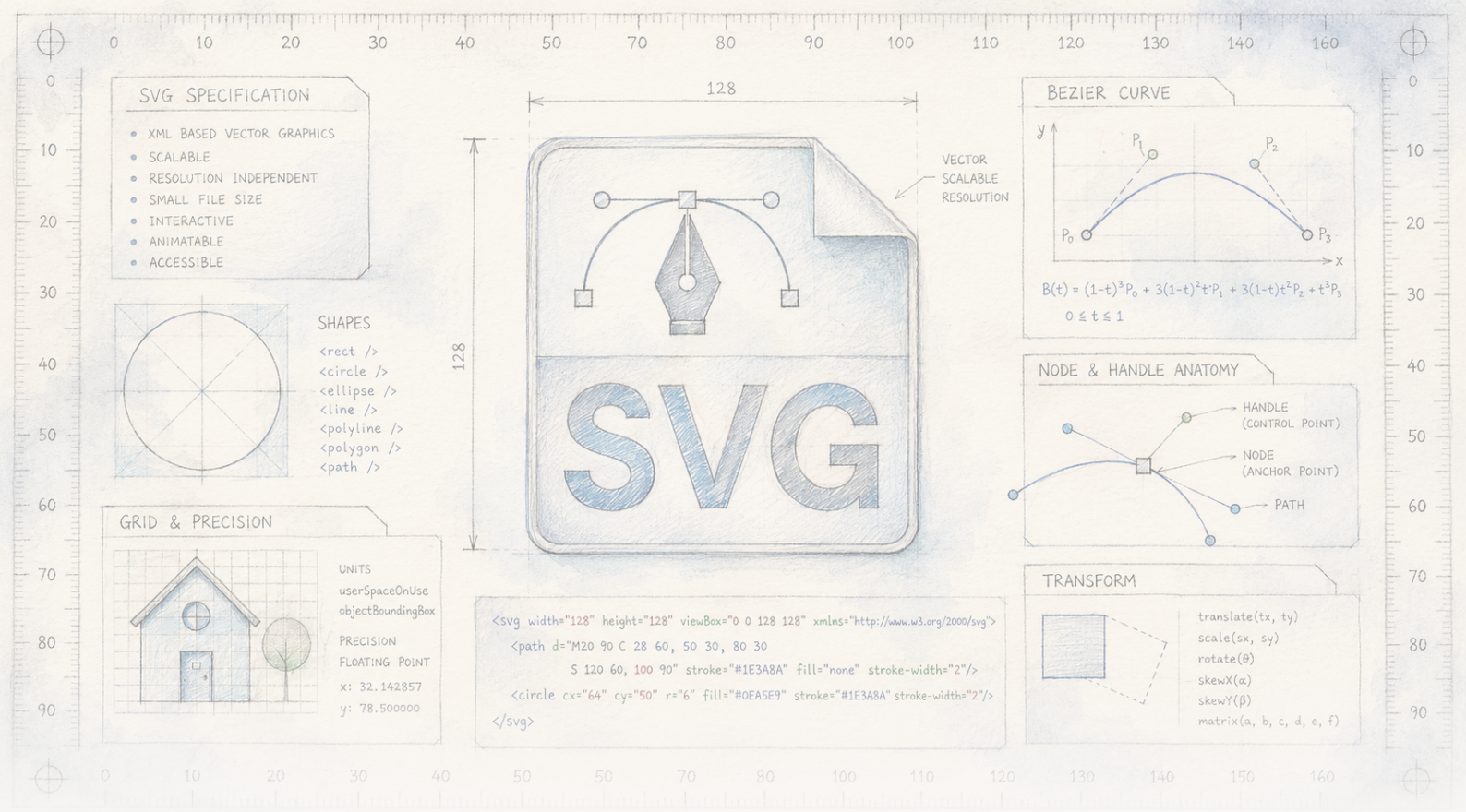


The Clean Paper

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

Kako naučiti risarsko umetno inteligenco, da pogleda na stran

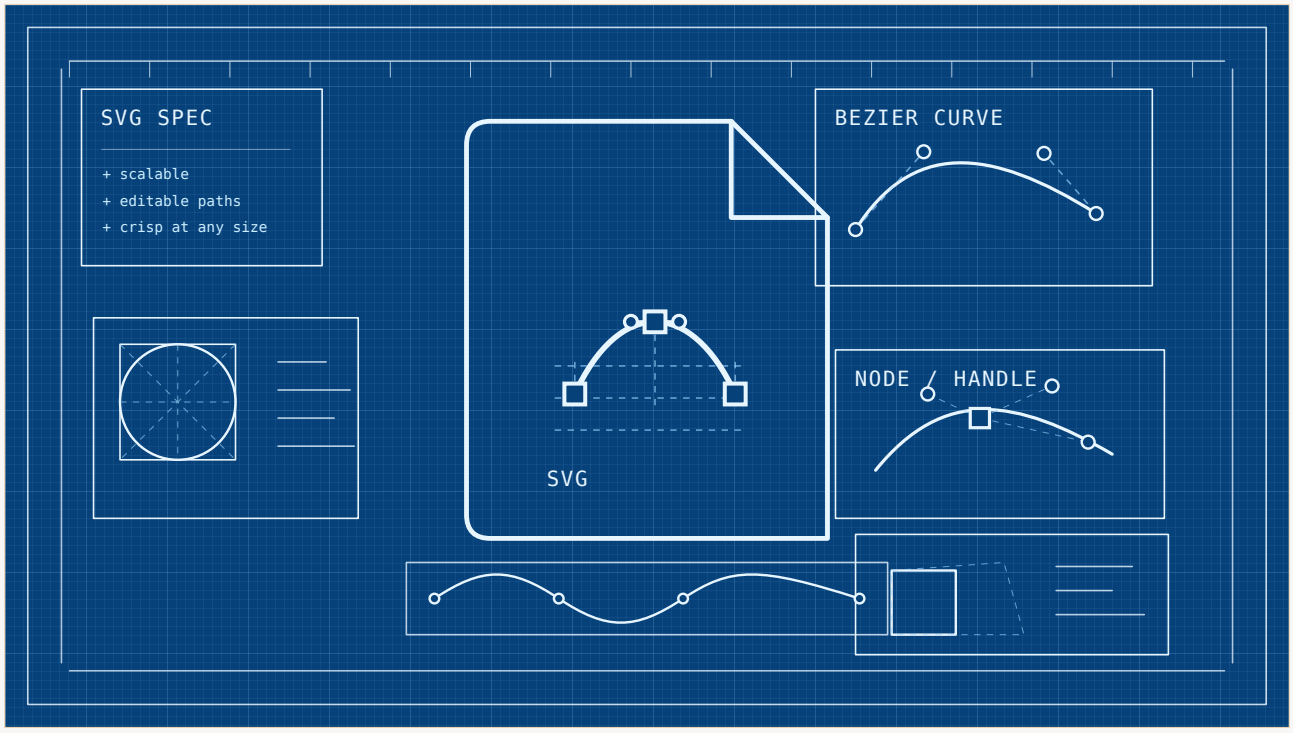
Jezikovni modeli, ki ustvarjajo vektorsko grafiko, to navadno počnejo na slepo — izpisujejo ukaze za risanje, ne da bi videli rezultat. Nova metoda modelu omogoči, da korak za korakom opazuje, kako nastaja platno, pošteni preobrat pa je, da samo dodajanje »oči« kakovost poslabša: model se mora naučiti, kako jih uporabljati. Rezultat je model, ki na enem benchmarku doseže ali rahlo preseže tekmece, učene z do dvajsetkrat več podatki — resničen in previden rezultat, ne revolucija.

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)

Risanje z zaprtimi očmi

Če današnjemu modelu AI naročite, naj nekaj nariše, se v ozadju lahko zgodita dve precej različni stvari. Znani generatorji slik — tisti, ki iz stavka ustvarijo fotografijo — rišejo neposredno v pikslih in med delom »vidijo« platno. Obstaja pa še druga, tišja vrsta risanja, pri kateri model sploh ne slika. Piše navodila: *tukaj nariši krog, tam črto, to obliko zapolni modro*. Ta navodila so koda — isti Scalable Vector Graphics oziroma SVG, ki stoji za večino ikon in logotipov na spletu — in prednost je resnična: rezultat ni fiksna mreža pikslov, temveč niz oblik, ki jih je mogoče poljubno povečati, prebarvati in urejati brez zameglitve.



Izvirna grafika SVG v slogu tehničnega načrta: sama slika je urejevalna vektorska datoteka, sestavljena iz poti, mrežnih črt, vozlišč in ročajev, ne iz fiksnih pikslov.

Laura Nesso / The Clean Paper Permission on file

Težava je bila, da je model, ki je pisal takšno risarsko kodo, do nedavnega delal na slepo. Celotno zaporedje navodil je izpisal v enem prehodu — krog, črta, zapolnitev — ne da bi jih vmes upodobil in preveril rezultat. Predstavljajte si, da rišete obraz z zaprtimi očmi: približno veste, kam sodijo oči, ne morete pa opaziti, da je druga pristala na licu ali da oblika las zdaj prekriva nos. Približno tako so delovali ti modeli, zato je bila koda lahko povsem veljavna, slika pa vizualno zmedena.

Ekipa pod vodstvom Guotaa Lianga je naredila skoraj komično očitno stvar: modelu je odprla oči. Njihova metoda *Render-in-the-Loop* po vsakem koraku upodobi nedokončano risbo in sliko vrne modelu, preden ta nariše naslednjo potezo. Resnično zanimivo pa ni to, da pomaga — temveč kaj je bilo potrebno, da je sploh začelo pomagati.

Kako oceniti risbo?

Ko članek napiše, da en model »premaga« drugega, je pošteno vprašati: pri čem in kako izmerjeno? Ocenjevanje generirane slike je res težavno — za »nariši prenosnik« ni ene same pravilne rešitve — zato področje uporablja nekaj samodejnih metrik, od katerih je vsaka le nepopoln približek človeškega pogleda.
V članku se pojavlja več takih mer. **FID** primerja splošne statistične lastnosti celotne serije generiranih slik z resničnimi; nižji rezultat je boljši, vendar opisuje zbirko, ne posamezne slike. **CLIP score** vpraša drug model AI, ali se slika ujema z besedilom poziva — uporabno, vendar le toliko dobro kot ta sodnik. **DINO**, **SSIM** in **LPIPS** primerjajo rekonstruirano sliko s ciljno, od surovih pikslov do naučenih značilk.
Nobena od teh metrik ni resnica. So nadomestne meritve in pogosto se premaknejo le za malo. Če en model doseže 127,6, drugi pa 128,8, je razlika resnična in v želeni smeri — vendar je to majhen premik, ne plaz, in še to v številki, ki le približno sledi temu, kar bi povedalo človeško oko. To je vredno imeti v mislih ob naslovu »premaga model, učen z dvajsetkrat več podatki«.

Kaj so naredili avtorji

Problem, ki so ga želeli popraviti, je prav slepo risanje. Obstoječi modeli obravnavajo pisanje SVG kot čisto besedilno nalogo: napovej naslednji kos kode iz dosedanje kode in je nikoli ne upodobi. Tako ostajajo neizkoriščene močne »oči« — vidni kodirnik — ki jih sodobni multimodalni modeli že imajo. Render-in-the-Loop nalogo preoblikuje v zaporedni vizualni proces. Po vsakem fragmentu kode se delni SVG upodobi v sliko in vrne modelu, tako da ta naslednji fragment izbere ob dejanskem pogledu na trenutno platno.

Prva ugotovitev je opozorilna in avtorji jo pošteno objavijo: če to zanko samo priključimo obstoječemu modelu brez posebnega učenja, ne deluje. Ko so vmesne upodobitve podali močnim splošnim modelom brez dodatnega treninga, se kakovost ni izboljšala — *poslabšala se je* pri vseh. Model, ki se ni naučil uporabljati svojih oči za to nalogo, tega ne zna nenadoma sam od sebe.

Večina dela je zato v učenju. Podatke za učenje so predelali tako, da je vsaka risba razbita na več majhnih, vizualno smiselnih korakov — kompleksne oblike razdelijo na preprostejše dele, da je v vsaki fazi mogoče videti nekaj novega — nato pa na teh zaporedjih dodatno učijo odprti model z osmimi milijardami parametrov, zgrajen na Qwen3-VL. To imenujejo Visual Self-Feedback. Dodali so še drugi mehanizem med samim risanjem, Render-and-Verify: preden sprejme novo potezo, jo model upodobi in preveri, ali je sliko dejansko spremenila ali samo ponovila prejšnjo. Poteze brez učinka zavrže, ko nič več ne pomaga, pa se ustavi. Vse to uporablja razmeroma majhen nabor — približno 850.000 primerov, manj kot polovico podatkov enega tekmeca in majhen delež podatkov drugega.

Kaj so ugotovili

- Tako učen model riše bolje od svoje slepe različice — najbolj očitno pri neuspehih, kjer slepi model postavi oko na lice ali namesto zahtevanega stolpčnega grafa nariše splošen zaslon.
- Na standardnem benchmarku MMSVGBench je metoda konkurenčna močnim tekmecem in jih pri več merah rahlo preseže — tudi OmniSVG, učen na več kot dvakrat več podatkih, in InternSVG, učen na približno dvajsetkrat več podatkih.
- Razlike so majhne. Na zbirki ikon je na primer glavni rezultat kakovosti slike 127,6 proti 128,8 pri najboljšem tekmecu, rezultat ujemanja s promptom pa 0,293 proti 0,291 — resnično in konsistentno, vendar ozko.
- Obe dodani komponenti prispevata. Če odstranimo posebno učenje ali preverjanje med risanjem, meritve opazno padejo. Zlasti preverjanje preprečuje, da bi se model zataknil pri vedno istem ponovnem risanju.
- Rezultat, ki ga avtorji sami poudarjajo, je učinkovitost — tako daleč pridejo z bistveno manj učnimi podatki kot vodilni modeli.

Česa to ne dokazuje

- **Ne** kaže, da je dovoliti modelu »videti« brezplačna zmaga. Ravno nasprotno: lasten poskus članka pokaže, da vizualna povratna informacija *brez ponovnega učenja kakovost poslabša*. Dobitek pride iz učenja, ne iz samih oči.
- **Ne** dokazuje velike ali odločilne prednosti. Pri večini mer je metoda zelo blizu tekmecem; »premaga model z 20× podatki« drži, vendar z majhnimi premiki nadomestnih metrik na enem benchmarku.
- **Ne** kaže splošne umetniške zmožnosti. Gre za raziskovalni model z osmimi milijardami parametrov, ki riše ikone in preproste ilustracije pri majhni fiksni ločljivosti 224×224 pikslov, ne za splošnega oblikovalca.
- Primerjava z velikim splošnim modelom GPT-5 **ni** enakovredna: ta model ni izdelan ali prilagojen za ozko nalogo risanja kode, zato zmaga tukaj zelo malo pove o obeh modelih na splošno.
- Metoda **ni** brez stroška. Upodabljanje in ponovno branje platna po vsakem koraku upočasni generiranje v primerjavi z enim slepim izpisom kode — strošek, ki ga avtorji priznavajo.

Kako močni so dokazi?

- **Trdni** so tam, kjer gre za nadzorovano primerjavo v lastnih pogojih. Ablacijski poskusi so čisti: odstranimo učenje ali preverjanje in rezultati padejo — obe sestavini torej res opravljata delo, ki jima ga avtorji pripisujejo.
- **Pošteni glede presenečenja.** Ugotovitev, da naivna vizualna povratna informacija *škodi*, ni skrita; je najzanimivejši del članka in uporaben popravek intuicije, da je več vhodnih informacij vedno bolje.

- **Tanki pri trditvi z lestvice.** Prednosti pred tekmeci z več podatki so majhne in temeljijo na enem benchmarku, ki so ga zgradili avtorji enega od teh tekmecev. Majhne razlike na nadomestnih metrikah enega testnega nabora so namig, ne dokončen odgovor.
- **Nepreizkušeni pri obsegu in v resnični uporabi.** Vse tukaj so ikone in preproste ilustracije nizke ločljivosti. Ali zamisel deluje pri zapletenih, visokoločljivostnih in praktičnih oblikovalskih nalogah, ostaja prihodnje delo — avtorji to jasno povedo.

Zakaj je pomembno

Osrednja ideja je skoraj smešno preprosta in sega daleč onkraj risanja. Če program nekaj ustvarja s pisanjem kode — spletno stran, graf, diagram, 3D-prizor — lahko vse napiše na slepo in upa, lahko pa sproti upodablja rezultat in popravi smer. Človek takšno zanko zapira avtomatično; na stran gledamo ves čas. Za te modele je to presenetljivo nova poteza.

Članek je vreden branja zaradi zvezdice, ki jo doda ideji. Omogočiti modelu, da vidi, ni isto kot ga naučiti gledati. Oči je treba naučiti uporabljati in šele takrat se zanka izplača — pa še takrat je rezultat resničen, a zmeren: bolj zanesljiv risar, ne nova vrsta umetnika. Za ljudi, ki gradijo orodja, kjer opis postane urejevalna grafika — na primer ikone in ilustracije aplikacij — je tiha praktična lekcija pomembnejša od lestvice. Zmaga je prišla iz cenejšega, pametnejšega učenja, ne iz več podatkov. To je uporabnejši sklep kot še ena točka na leaderboardu.

Kratek povzetek

Jezikovni modeli, ki ustvarjajo vektorsko grafiko — urejevalno in poljubno povečljivo kodo za večino spletnih ikon in logotipov — so tradicionalno delali »na slepo«, saj so izpisali vse ukaze za risanje, ne da bi vmes upodobili rezultat. Ekipa pod vodstvom Guotaa Lianga predlaga Render-in-the-Loop: po vsakem koraku se nedokončana risba upodobi in vrne modelu, tako da naslednjo potezo naredi ob pogledu na platno. Osrednja in poštena ugotovitev je, da samo dodajanje te možnosti obstoječemu modelu rezultat *poslabša*; izboljšanje pride šele, ko se model na vizualno povratno informacijo posebej nauči, pri tem pa pomaga še preverjanje, ki zavrne poteze brez učinka. Tako ponovno učen model z osmimi milijardami parametrov na standardnem benchmarku dosega ali rahlo presega tekmece, učene z do dvajsetkrat več podatki — resničen rezultat, vendar z majhnimi razlikami na nadomestnih metrikah, za ikone in preproste ilustracije nizke ločljivosti. Avtorji poudarjajo učinkovitost: dobro gledanje lastnega dela lahko delno nadomesti samo količino učnih podatkov.

Preverjanje brez olepševanja

Kaj članek pokaže: Ponovno učenje modela za vektorsko grafiko, da po vsakem koraku upodobi in pogleda nedokončano risbo, daje boljše in popolnejše rezultate kot slepo risanje — na enem standard-

nem benchmarku je konkurenčno in rahlo boljše od tekmecev z več učnimi podatki.

Kaj je verjetno, vendar ne dokazano: Da je »gledanje lastnega dela« splošno boljši recept kot samo več podatkov; da bo zamisel pomagala pri zapletenih visokoločljivostnih ali resničnih grafičnih nalogah; da majhne razlike na benchmarku predstavljajo razliko, ki bi jo človek dejansko opazil.

Česa ne pokaže: Da vizualna povratna informacija pomaga sama po sebi — brez ponovnega učenja škodi; da je prednost pred tekmeci velika ali odločilna; splošne risarske zmožnosti; poštene neposredne primerjave s splošnimi modeli, kot je GPT-5, ki niso zgrajeni za to nalogo.

Glavne omejitve: En benchmark, delno zgrajen pri avtorjih tekmeca; majhne razlike na nadomestnih metrikah; 8B model, ikone in preproste ilustracije pri 224×224; počasnejše generiranje zaradi upodabljanja po vsakem koraku.

Koliko zaupanja naj ima splošni bralec? Visoko, da zapiranje zanke — sprotno upodabljanje in ponovno gledanje — res pomaga in da se mora model tega naučiti, ne le dobiti funkcijo. Nizko do zmerno, da ta konkretni model odločilno presega tekmece; »premaga model z 20× podatki« je resnična, vendar skromna ugotovitev z enega benchmarka. Visoko v eno idejo, ki si jo je vredno zapomniti: pri kodi, ki riše, je gledati sproti bolje kot risati na slepo.

Viri

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/