LAYOUT BEYOND FORMAT: INTERNALIZING SPATIAL EXECUTION IN LLMS VIA ELEMENT-LEVEL GEOMETRIC REWARDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models generate visual code—SVG, HTML, slide scripts—that is almost always syntactically valid, yet the rendered layouts frequently exhibit spatial defects; existing systems compensate through repeated inference-time detection and rewriting, at a cost that accumulates with every call. We observe that models do not lack spatial judgment: they reliably identify and follow spatial constraints, yet fail to maintain them during autoregressive generation. This local planning failure lies in the decoding process itself rather than in understanding, long context, or format syntax, and since the cause is format-agnostic, layout ability should be acquired once, at the format-agnostic level. We therefore build a two-stage framework that first generates a format-agnostic layout skeleton and then realizes it into code of any target format, making planning an independently supervisable object, and post-train on top of it: SFT distills two-stage trajectories, and RL uses a verifiable dense reward from a geometric detector that attributes every defect to the tokens that produced it, alongside a page-level outcome reward. A 9B model trained this way raises its end-to-end delivery rate from 25.6% to 85.6%; the two-stage formulation on its own, without training, does not help—it lowers the same model to 17.1%—so the gain comes from internalizing the geometric signal into the weights rather than from the scaffold. As the diagnosis predicts, the resulting layout ability transfers zero-shot to code formats absent from all training data and improves quality and efficiency inside an established agentic pipeline.

1 INTRODUCTION

Frontier large language models now generate visual code—SVG graphics, HTML pages, slide scripts—that is almost always syntactically valid, yet rarely spatially correct: the rendered pages overlap, overflow, and misalign. Because people read these artifacts directly, layout defects decide their usability even when the content is right. In our evaluation, frontier models deliver a clean page on only 15–35% of tasks in a single call, and smaller open-source models lag further behind.

The prevailing remedy is inference-time engineering: constrain generation with harnesses or skill pipelines, then repair the output through repeated review. This works, but the cost is paid anew at every call, and end-to-end metrics cannot attribute a gain to the model or to the framework. Meanwhile SVG, web-page, and slide generation are optimized as separate tasks, even though the decisions that matter—what content goes where, at what hierarchy and spacing—are largely shared, echoing evidence that models carry format-agnostic internal representations beneath divergent surface syntaxes (Yuan & Su, 2026); training on a single format forfeits that shared ability. Existing training attempts instead score rendered images with visual models (Rodriguez et al., 2025b; Chen et al., 2025c; Jiang et al., 2026; Liu et al., 2026a), which miss the subtle, continuous, precisely localizable geometric errors of layout.

Controlled experiments locate the failure precisely (Appendix A.1). Models do not lack geometric understanding—they identify layout defects in their own finished outputs and follow externally declared spatial constraints almost perfectly, the two abilities engineering schemes rely on—nor do they lose long-range context: conflicts occur almost exclusively among the few most recently

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

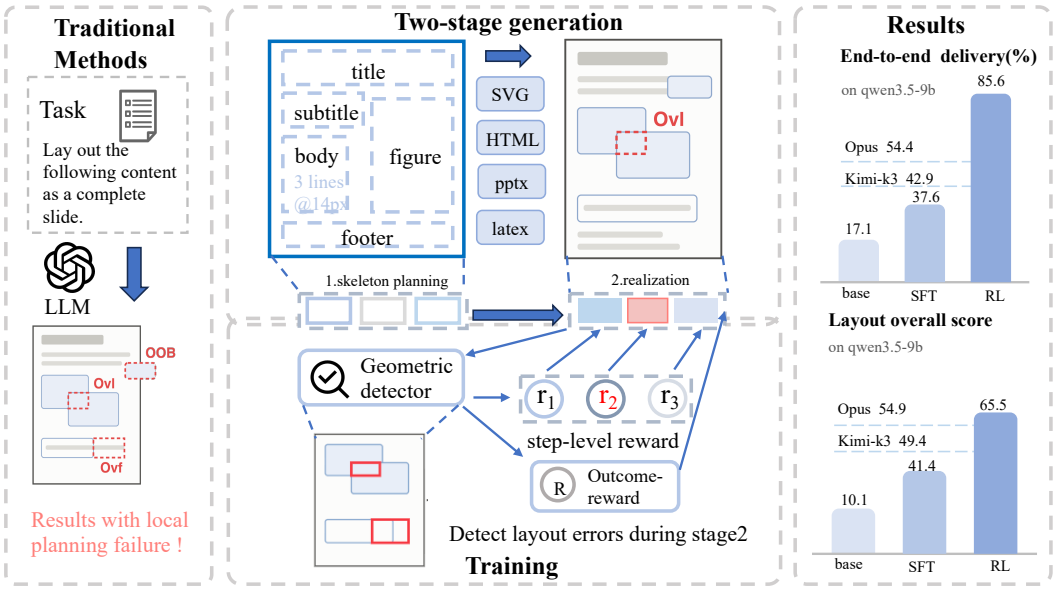


Figure 1: Overview. **Left**: one-shot generation exhibits *local planning failure*—pages that are syntactically valid yet carry overlap (Ovl), overflow (Ovf), and out-of-bounds (OOB) defects. **Middle**: we factor generation into a format-agnostic skeleton-planning stage and a realization stage, and internalize the detector's verification into training: element-level step rewards attribute each defect to the element that produced it, alongside a page-level outcome reward. **Right**: on Qwen3.5-9B, delivery and the composite layout score rise from base through SFT to RL, surpassing the frontier backends.

generated elements (median memory distance 3–4). The fault is in decoding itself: the model fails to maintain the relative positions of what it has just produced. We call this *local planning failure*. Two properties follow. Errors are attributable—each defect traces to the span that emitted it. And verification is not scarce: the model can judge its own output after the fact, and a geometric detector localizes every defect exactly; what is scarce is where it is paid for. Scaffolding pays at every call; we pay once, at training time.

Because this failure belongs to decoding rather than to any format's syntax, it predicts that planning ability trained on one format transfers to others. Our design follows (Figure 1): a two-stage architecture generates a layout skeleton, then realizes it in the target syntax, making planning an independently supervisable target while realization carries only syntax. We measure this ability with a unified cross-format evaluation suite whose primary measure is end-to-end delivery rate. SFT distills two-stage trajectories from a strong teacher; RL then turns the detector's per-defect attribution into a dense, verifiable reward that internalizes spatial maintenance during generation.

With this training, a 9B open-source model raises its end-to-end delivery rate from 25.6% to 85.6%. The prediction is borne out: planning gains transfer zero-shot to unseen code formats and carry into an established agentic pipeline, improving quality and efficiency—confirming that what is learned is a model-level, format-agnostic ability rather than overfitting to one task.

Our main contributions are:

- **A format-agnostic two-stage generation framework.** It decouples the layout planning shared across formats from format-specific syntax realization, making planning an explicit, independently supervisable, and format-reusable training target.
- **A training recipe centered on verifiable geometric rewards.** SFT distills complete two-stage trajectories from a strong teacher; our RL stage attributes every geometric defect to the token span that produced it, combining element-level step rewards with page-level outcome rewards. Existing verifiable-reward training for layout scores geometry only as a page-level scalar shared by the whole rollout (Pan et al., 2026; Li et al., 2026b; 2025); to our knowledge, ours is the first whose geometric reward is dense at the element level.

- **Evidence of model-level generalization.** Under a unified evaluation, crossing backends with inference scaffolds separates the model's contribution from the harness's, showing the gains belong to the model's own layout ability: they transfer zero-shot to code formats absent from all training data, and persist from a single plain call to an agentic slide-generation pipeline, where the trained model both raises delivery quality and cuts review-and-rewrite rounds.

2 Related Work

2.1 Visual Code Generation and Layout Models

Image-generation tasks such as posters, documents, and slides have shifted from diffusion models to LLMs emitting renderable code, along separate lines for SVG (Rodriguez et al., 2025a; Xing et al., 2025; Chen et al., 2025a;b; Zou et al., 2024), HTML (Si et al., 2025; Yun et al., 2024; Gui et al., 2025a; Wan et al., 2025; Gui et al., 2025b), slides (Zheng et al., 2025b; Tang et al., 2025), charts (Yang et al., 2025b; Wu et al., 2024), and posters (Zhang et al., 2026b), judged by how closely the render matches its target, with layout noted only in passing. The failures reported are nonetheless the same everywhere—occlusion, overflow, out-of-bounds placement, broken alignment—each format explaining its own. A second line generates labeled bounding boxes (Inoue et al., 2023; Zhang et al., 2023; Chen et al., 2024; Horita et al., 2024; Weng et al., 2024; Lin et al., 2023a), increasingly with LLMs (Tang et al., 2024; Feng et al., 2023; Lin et al., 2023b; Shi et al., 2025; Yang et al., 2024; Seol et al., 2024), with better geometry but a fixed canvas, predefined element categories, and abstract boxes rather than renderable code; placed before code generation it adds an external constraint without changing how the model generates. We treat layout as an ability of the model itself, to be diagnosed and trained.

2.2 Feedback Loops and Their Modality

Without seeing its own render a model's layouts are weak and degrade as elements accumulate (Qiu et al., 2024; Chen et al., 2025b), and models do not revisit such errors unprompted (Tsui, 2025; Jiang et al., 2025; Maimon et al., 2026). The common remedy adds a render–check–repair loop, since checking is easier than generating (Chen et al., 2025c); these works differ only in when the check is paid for—at inference time (Liang et al., 2026; Wang et al., 2026; Zhang et al., 2026b; Gui et al., 2025b) or as a training reward (Rodriguez et al., 2025b; Chen et al., 2025c; Liu et al., 2026a; Xu et al., 2026b; Liu et al., 2026b)—and agree on its source and resolution: the render, read by a visual model, reported for a whole page or round. Three problems follow. Optimization amplifies the judge's own preferences (Mao et al., 2026; Yao et al., 2026) and collapses within-group variance (Sun et al., 2026); visual judges cannot say which element conflicts with which (Zhang et al., 2026a), so there is nothing to attribute; and the reward exists only when the render does, so in unforgiving formats a syntax slip voids the whole rollout—most often for the weakest models, whose failed prefixes remain largely usable.

2.3 Reinforcement Learning and Reward Design for Layout

Layout rewards differ in source: some use geometric quantities computed from the layout (Pan et al., 2026; Wei et al., 2025; Li et al., 2026b), others score the render with a visual model or mix the two (Xing et al., 2025; Rodriguez et al., 2025b; Li et al., 2025; Ananda Kumar & Arunachalam, 2026; Jiang et al., 2026; Xu et al., 2026a). The purely geometric route has a measured ceiling—bounding-box metrics alone halve overlap but fall short of methods with a visual encoder (Li et al., 2026a; Seol et al., 2024)—so geometry does not reach aesthetics or semantic fit; we build on it for four other properties: it cannot be fabricated, can be recomputed and checked, points to a specific element, and survives artifacts that do not render. In granularity, all such rewards hand one scalar to an entire trajectory or round, without distinguishing which placement caused the defect. That is costly here: most elements are placed correctly and the problems sit in a few, so a per-round return spreads credit and blame evenly over every step. To our knowledge no prior work traces a defect to the responsible element and then to the token span that wrote it, which is what our training signal rests on.

3 PRELIMINARY: PROBLEM FORMULATION

Given source content D and a canvas C, visual code generation requires the model to autoregressively emit a sequence of elements $s_1, \ldots, s_N$, where each element $s_i = (g_i, c_i)$ carries geometric attributes (a bounding box) and content attributes (text, style); a valid layout requires every pair of elements to satisfy a spatial constraint set $\mathcal{R}(s_i, s_j)$ covering non-overlap, alignment, spacing, and boundary containment. Generation thus forms a Markov decision process: the state at step t is $S_t = (C, D, s_{1:t})$, and the policy $\pi(\cdot \mid S_t)$ emits the next element.

Under this formulation, our controlled probes characterize how LLMs fail at this task (experimental design and full analyses in Appendix A.1): (i) the verification side is intact—models identify pairs violating $\mathcal{R}$ post hoc and follow externally given spatial constraints; (ii) the failure sits at policy sampling—violated pairs concentrate among near neighbors in sequence order (median distance 3–4 elements), independent of canvas occupancy, ruling out context truncation; (iii) the process has no backtracking—later actions condition on the erroneous prefix and misplaced elements become references for subsequent ones, so layouts are sticky—an anchoring phenomenon also observed in non-autoregressive layout generators (Iwai et al., 2024)—and post-hoc whole-page repair is costly and unreliable. These properties directly inform the designs that follow.

4 METHOD

Each property prescribes a design. Property (ii)—the model cannot solve the joint constraint problem while emitting—says to lift planning out of the per-step decision. Properties (i) and (iii)—verification is intact, and a defect left in place anchors later elements—say to run the detector during generation and rewrite offenders in place. At training time the same attributability turns the detector's signal into a dense per-element reward rather than one page-level score. The three subsections below take these in order (Figure 2).

4.1 TWO-STAGE GENERATION

Property (ii) locates the failure at sampling time: the policy cannot solve the joint constraint-satisfaction problem while emitting. We therefore lift planning out of the per-step decision. The model first produces an explicit global plan P—a format-agnostic layout skeleton—then realizes it under the target format's syntax, conditioning each step on $S_t = (C, D, P, s_{1:t})$; every step reduces from planning-plus-rendering to rendering alone.

4.2 STEPWISE DETECT-AND-REWRITE

Verification must therefore be interleaved into the transitions rather than applied afterwards: layouts are sticky, so a misplaced element must be caught before later ones anchor to it. Since verification is intact and cheap, a geometric detector supplies exactly this signal: our stepwise mechanism checks each batch of newly generated elements, names the defective ones, and has the model rewrite them locally before generation proceeds. Appendix A.2 verifies its effectiveness, which motivated the RL step reward.

4.3 TRAINING

Layout defects are state-transition-level events: every violation is attributable to the action—the token span—that produced it. Our pipeline exploits this, distilling the two-stage policy via SFT and then internalizing the detector's signal via RL.

Supervised fine-tuning. SFT equips the base model with the two-stage policy: plan a full page as a well-formed skeleton, then realize it in target-format code without deviating. We generate trajectories with a strong teacher on corpora disjoint from evaluation and keep one only if it passes four gates: the skeleton is well-formed (parseable, budget-consistent) and covers the source content, and the realized page renders defect-free and clears the coverage gate.

Reinforcement learning. RL internalizes external verification so the model detects and corrects defects on its own. Rollouts mirror the two-stage task—a skeleton, then a realization—with policy gradients on the realization turn. The centerpiece is an **element-level step reward**—the training-time counterpart of stepwise rewriting, and the per-transition supervision the MDP view calls for. Local planning failure is a **per-step pathology**: page-level scores, like prior visual-modality rewards, collapse every transition into one scalar and leave the model to guess which placement was at fault. We instead **attribute every defect to the offending element**, and hence to the token span that emitted it. For element i on span $\mathcal{T}_i$ the detector returns defect records $\mathcal{V}_i$, aggregated by family and severity:

$$r_i \;=\; \max\Big(r_{\min},\; \sum_{v\in\mathcal{V}_i} w_{f(v)}\,\rho_{f(v)}\,\sigma(v)\Big), \qquad d_t \;=\; \frac{r_i}{|\mathcal{T}_i|} \quad \text{for } t\in\mathcal{T}_i, \tag{1}$$

where $f(v)$ is the record's defect family, $w_f < 0$ the weight of that family, and $\sigma(v) \in (0, 1]$ a severity normalized against the detector threshold that fired—so that weights remain comparable across families. The factor ρ_f halves the families whose records name a pair of elements and are therefore charged to both endpoints, and $r_{\min}$ clamps one element's total so that no single placement can dominate a rollout. Spreading r_i evenly over the element's own tokens makes its contribution independent of how verbosely it was written; tokens belonging to no element receive $d_t = 0$ (family weights, per-family severities, and calibration in the appendix). The reward thus lands exactly **where the failure occurs**: at the emission steps whose spatial relations it teaches the model to maintain.

The step reward alone, however, admits a degenerate optimum—a model can avoid penalties by **drawing less**. As a safeguard against overfitting to local geometry at the expense of overall delivery, we add an **auxiliary** page-level **outcome reward**,

$$R \;=\; g(c)\,\big(\lambda_c\, c + \lambda_e\, e^{-D/\kappa} + \lambda_{\text{fill}}\,\text{fill}\big), \tag{2}$$

where, D is the total defect count, fill measures how fully the page is populated, and $g(\cdot)$ is a soft coverage gate; the gate and the fill term ensure a page is rewarded only when its content is actually delivered. Its role is **regularization**: anchoring holistic quality while the step reward does the targeted work. The two combine in the advantage,

$$A_t \;=\; \hat{A} + \lambda\, d_t, \qquad \hat{A} \;=\; \frac{R-\mu_R}{\sigma_R}, \tag{3}$$

where the dense term $\lambda\, d_t$ carries the element-level signal, concentrating the penalty on the exact tokens that produced each defect, and $\hat{A}$ is the outcome advantage of GRPO (Shao et al., 2024), normalized over a group of rollouts of the same prompt and shared by all tokens of a rollout.

5 Experiment

5.1 Evaluation

We sample from open-source datasets released by prior work and recast every instance into a single uniform task: generating visualization code from text. To keep the suite diverse and demanding, the source data span SVG, slides, and HTML. Where a dataset supplies only images as input, we extract the underlying text content and task description with Opus 5 or with web and slide parsing tools. The sampled datasets are WebCode2M (Gui et al., 2025a), PPTAgent (Zheng et al., 2025b), `pptx_templates` (noxneural, 2024), Slide2Code (Tang et al., 2025), Zenodo10K (Zheng et al., 2025a), and Design2Code (Si et al., 2025), together with a proprietary corpus of clinical medical text.

We measure four aspects of the output. Layout quality is measured primarily by the incidence of the five defect families common to these tasks. Delivery is reported as a funnel whose headline figure is the end-to-end delivery rate, the fraction of all tasks yielding a clean page, factored into a completion rate, a gate rate, and a conditional defect-free rate. Content fidelity is measured by content coverage and declaration compliance, and aesthetics by element count, whitespace ratio, balance, and alignment. All of these are computed by external scripts and involve no visual-modality judgment. To corroborate them we additionally report two existing protocols, PPTEval (Zheng et al., 2025b) and WebGen-Bench (Lu et al., 2025), which score rendered outputs along several dimensions with a multimodal model. Full details are in Appendix A.3.

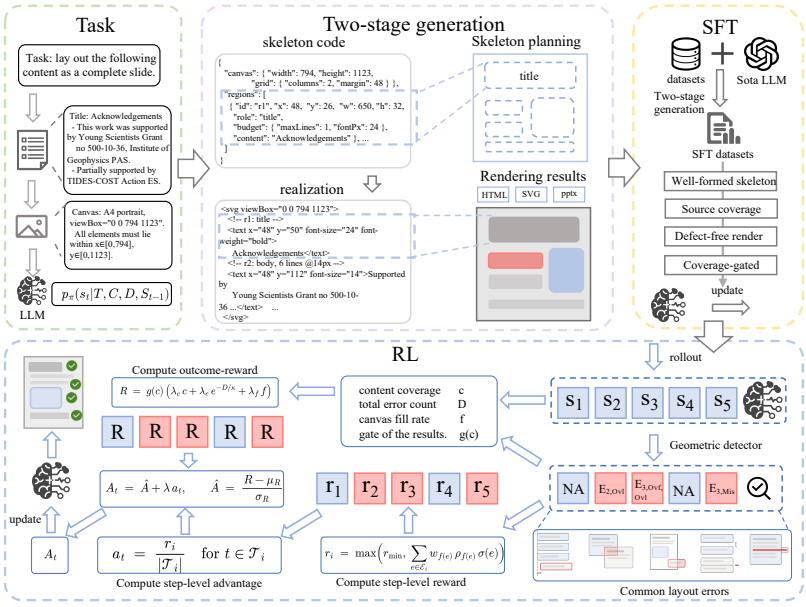


Figure 2: The training pipeline. A layout task is answered in two stages: skeleton planning, then realization into the target format. SFT distills two-stage trajectories from a strong teacher, keeping only those that pass four gates (well-formed skeleton, source coverage, defect-free render, coverage-gated). RL rolls out the two-stage policy; the detector attributes each defect to the offending element, and the element-level step reward (Eq. 1) combines with the page-level outcome reward (Eq. 2) in the advantage (Eq. 3).

5.2 Training

Balancing capability against size, we take Qwen3.5-9B and Qwen3.5-4B as our base models. We sample instances from WebCode2M-train (Gui et al., 2025a), MTSamples (Boyle, 2018), Zenodo10K (Zheng et al., 2025a), Slide2Code (Tang et al., 2025), and `pptx_templates` (noxneural, 2024), and obtain targets by prompting Opus 5 under our two-stage procedure. We then filter the trajectories, discarding those whose first-stage skeleton is ill-formed and those whose second-stage output carries layout defects; the survivors are the cold-start data for SFT, and the discarded instances form the prompt pool for RL. Further details are in Appendix A.4.

5.3 Main Results

As Table 1 shows, both trained models far exceed the base model and match the frontier backends (Opus 5, Kimi-K3 (Moonshot AI, 2026), Qwen3.8-27B): on defect-free rate, content fidelity, and aesthetics they approach frontier quality. Both stages contribute, and the gain from RL is the larger: on defect-free rate it far surpasses both SFT and the base model.

This matches the design intent. The SFT cold start teaches instruction following under the two-stage paradigm—writing a well-formed skeleton and filling it in as declared—and the RL stage then internalizes, through the detector's element-level step reward, the maintenance of relative positions among generated elements. The final gain thus comes primarily from RL rather than from the teacher's distillation data. Training is also stable: the model does not evade defects by drawing less—RL draws fewer elements than SFT (48 vs. 63) yet reaches the highest fill-gated composite score of any arm (LQS 65.5, Table 17).

The reward and the reported metrics are computed by the same detector, so we checked whether the gains are an artifact of that instrument. Three checks, each independent of the training signal, say they are not. First, the reward and the metrics are different objects rather than one number read twice: the reward charges the token span that emitted each defect and passes through a coverage gate and a fill term, while the metrics are page-level statistics of a fixed, unweighted count, so optimizing the former does not mechanically move the latter. Second, the two families of metric

Table 1: Main results on the SVG suite. Defect columns are counts per page; E2E is the end-to-end delivery rate over the full 450-task suite. The last two columns are judged rather than computed and rest on a smaller common intersection of tasks (Appendix A.3); L3-C is capped below 3 by its rubric and is not marked. Ne is a guardrail rather than a quality measure: since defect counts fall toward zero on sparse pages, the defect columns are read against it. The full metric set is in Appendix A.5. Best value per column is bold within each comparison group—the arms of one model, or the trained models at one scale—and undirected, reference-only, and guardrail columns are unmarked.

		Layout compliance						Delivery	Content	Aesthetics
Model	Inference	Ovl↓	Spc↓	Mis↓	Ovf↓	OOB↓	Ne	E2E %↑	L3-C	L3-D↑
Opus 5	one-shot	1.04	1.20	0.35	0.93	0.05	60	33.3	2.62	3.69
Opus 5	two-stage	**0.07**	**0.44**	**0.10**	**0.12**	**0.00**	57	**54.4**	2.10	**3.83**
Kimi-K3	one-shot	3.47	1.93	0.85	0.28	0.04	57	16.0	2.69	**3.86**
Kimi-K3	two-stage	**0.29**	**0.55**	**0.15**	**0.01**	**0.00**	41	**42.9**	2.07	3.79
Qwen3.8-27B	one-shot	2.03	1.09	0.44	0.43	0.05	54	**34.7**	**2.69**	3.69
Qwen3.8-27B	two-stage	**0.33**	**0.57**	**0.25**	**0.03**	**0.00**	39	31.3	2.17	**3.79**
Qwen3.5-4B	one-shot	**0.79**	**0.20**	**0.01**	**1.47**	0.30	27	**32.4**	1.55	**1.31**
Qwen3.5-4B	two-stage	1.88	0.70	0.02	1.66	**0.20**	25	10.9	1.34	1.17
Qwen3.5-9B	one-shot	0.93	**0.21**	**0.02**	2.17	0.46	32	**25.6**	1.59	1.38
Qwen3.5-9B	two-stage	**0.67**	0.68	**0.02**	**1.75**	**0.14**	28	17.1	1.52	**1.90**
9B-SFT	two-stage	0.34	0.49	0.15	0.45	0.03	63	37.6	1.66	3.34
9B-RL	two-stage	**0.01**	**0.03**	**0.00**	**0.06**	**0.00**	48	**85.6**	**1.79**	**3.38**
4B-SFT	two-stage	1.84	1.03	0.18	0.65	0.08	69	25.6	1.55	2.66
4B-RL	two-stage	**0.02**	**0.02**	0.01	**0.05**	**0.00**	44	**88.0**	1.69	**3.17**
4B-RL-GRPO	two-stage	0.07	0.04	0.00	0.07	0.00	42	77.1	1.61	3.07

do not move together—Opus 5 and Kimi-K3 gain substantially on every defect family under two-stage inference, yet their L3-C content scores fall (2.62 to 2.10 and 2.69 to 2.07); if the judged and geometric columns were two readings of one quantity, that inversion would not occur. Third, the gains extend beyond what the reward names: the L3-D design score comes from an external judge the reward never sees, the transfer formats appear in no training rollout, and acceptance rates come from a third-party pipeline we did not write. A reward cannot overfit a benchmark it never touches.

On a weak base the same paradigm is a net harm ($25.6\% \rightarrow 17.1\%$ on the 9B base): the model cannot write a well-formed skeleton, and 38.8–45.0% of its declared regions cannot geometrically hold their content, against 0.8–6.0% for the frontier backends (Table 19). After training it reaches 85.6%: what fails on the base is the ability the paradigm presupposes, not the paradigm.

5.4 Comparison With Traditional RL Systems

To isolate the value of the step reward, we compare against the conventional practice of driving GRPO with a single outcome score, on Qwen3.5-4B, mirroring prior methods such as AeSlides (Pan et al., 2026). Our outcome term is computed geometrically rather than multimodally: on weak models few rollouts render successfully early on, so discarding the rest wastes data, and multimodal scores are preference-biased and weakly discriminative (Mao et al., 2026). The two arms differ in one respect only: the dense arm adds the element-level step reward of Eq. 1, so any gap between them is attributable to the element-level signal rather than to the outcome term. Table 1 reports the comparison. The step reward is worth its cost on the measures that matter—it halves overlap ($0.07 \rightarrow 0.02$), halves spacing violations ($0.04 \rightarrow 0.02$), lowers overflow ($0.07 \rightarrow 0.05$), and lifts end-to-end delivery by almost eleven points ($77.1\% \rightarrow 88.0\%$), with both judge scores moving up as well—so the element-level signal is doing work that a single page-level scalar does not. One column favors the control: misalignment sits at 0.01 against 0.00, a difference of a single hundredth of a defect per page, on a family that both arms have already driven to the floor. We report it rather than round it away because it marks the shape of the finding: dense credit assignment buys large gains on the families where defects cluster, and is neutral where they are already absent.

Table 2: Zero-shot transfer to unseen code formats, measured on geometric intent (two-stage inference, 450 tasks per format). Ovl and OOB are counts per page recomputed from the coordinates declared in the code; Int-DF and Int-E2E are defined in the text; Align, Cov, and Decl follow Appendix A.3 on the same declared geometry. Ne is the element-count guardrail: on HTML the SFT row draws $2.8\times$ more elements at the same raw overlap, i.e. a $2.8\times$ lower rate per element. SFT is the strongest cross-format SFT configuration (one-line syntax note in the realization prompt); RL is the checkpoint of Table 1. Best value per column is bold within each comparison group; undirected columns are unmarked.

Format	Model	Ne	Ovl↓	OOB↓	Int-DF %↑	Int-E2E %↑	Align↑	Cov↑	Decl %↓
HTML	Base	9.6	2.99	0.92	21.1	16.0	0.963	0.707	39.4
HTML	SFT	26.5	3.00	**0.20**	**56.3**	**48.7**	0.988	**0.742**	**3.7**
HTML	RL	16.7	**0.57**	1.03	48.9	20.7	**0.995**	0.720	6.4
LaTeX	Base	4.4	0.75	0.74	57.1	44.7	0.985	0.623	38.6
LaTeX	SFT	13.6	0.42	0.14	90.1	**80.7**	0.993	0.681	**2.9**
LaTeX	RL	8.5	**0.05**	**0.02**	**97.0**	79.8	**0.999**	**0.694**	6.0
PPTX	Base	9.9	3.86	**0.48**	24.3	13.1	0.948	0.717	43.3
PPTX	SFT	28.8	2.94	0.56	**53.6**	**44.2**	0.988	**0.752**	**4.9**
PPTX	RL	23.8	**2.07**	2.06	27.2	23.8	**0.995**	0.738	7.0

5.5 Generalization Across Code Formats

Local planning failure belongs to decoding, not to any format's syntax, so planning ability trained on one format should carry to others. We test this on the three formats absent from all training data—HTML, LaTeX, and `python-pptx`, three unrelated syntax systems (CSS declarations, TeX macros, a Python API)—running the unchanged two-stage procedure on the full 450-task suite. Rendered metrics cannot answer the question alone: in an unfamiliar format a syntax slip (a unitless CSS length, an SVG attribute name inside TikZ) makes the renderer drop elements wholesale, so a page with correct coordinates can score as a pile of overlaps or not compile. We therefore also measure *geometric intent*: positions and sizes are read from the code text whether or not it renders, and Ovl and OOB are recomputed on those declared boxes. *Int-DF* is the fraction of coordinate-bearing products with both at zero; *Int-E2E* divides the same numerator by 450.

Table 2 shows a consistent trend in all three formats; the 4B line appears in Appendix A.7. Per-element overlap falls at every stage and is lowest under RL, by 4.5–$28.7\times$ against the base; alignment rises monotonically through SFT to RL, and coverage and declaration violations also improve. Since none of these formats appeared in training, the progression reflects abstract layout ability—stage one teaches planning, stage two internalizes maintaining relative positions—rather than any one dialect.

In places RL still trails SFT (Int-E2E on HTML, 20.7 against 48.7; OOB on PPTX, 2.06 against 0.56). This is a syntax effect, not a layout one: extensive training on one format crowds out the others' dialects in a model with limited base coding ability, most so for our smallest model. A stronger backbone would absorb the transfer more readily, and training directly on the target syntax is the evident way to close the gap.

5.6 Generalization within Existing Skills

To test whether the gain extends beyond our own task, we integrated the trained model and its inference paradigm into an established agentic slide-generation workflow (Table 3).

We run *ppt-master*, a common skill that has a model author SVG code later assembled into slide decks, and compare its original pipeline against variants that embed our trained models and the two-stage paradigm. On the base model two-stage inference is a net harm, exactly as in Section A.2; the trained models and the stronger frontier backends instead gain substantially under the same paradigm, and our model reaches and on several measures surpasses the frontier. *First-pass*—the probability a page clears the skill's internal inspection first try, an indirect measure of retries—improves as well, so the trained ability also raises efficiency inside the loop. This figure also speaks to the concern of Section 5.3 about the reward and the metrics sharing an instrument: the skill's inspection is not our detector, and a model that had merely learned to satisfy our thresholds would

Table 3: Generalization to existing skill. Model × inference paradigm on an agent workflow that plans, generates, and self-revises slide decks. Delivery measured as LQS, DF (final defect-free rate over scored pages), E2E, and first-pass acceptance (fraction of pages needing zero revisions), all in percent. Base = the untrained model at each scale. Best value per column is bold within each comparison group; undirected columns are unmarked.

Model	Inference	Ovl↓	Spc↓	Mis↓	Ovf↓	OOB↓	Ne	LQS↑	DF↑	E2E↑	First-pass↑
Qwen3.5-9B	one-shot	1.89	**0.29**	**0.01**	**1.46**	0.26	21.8	**12.8**	**25.0**	**19.4**	**55.6**
Qwen3.5-9B	two-stage	**1.73**	0.37	0.02	2.02	**0.16**	18.9	8.3	20.3	14.4	18.4
9B-SFT	one-shot	3.47	0.90	0.12	1.54	**0.25**	52.5	16.7	25.4	19.4	75.6
9B-SFT	two-stage	**1.01**	**0.12**	**0.11**	**0.36**	0.33	40.0	**42.8**	**49.7**	**42.8**	**95.6**
9B-RL	one-shot	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	16.1†	32.8	100.0†	97.2†	**98.3**
9B-RL	two-stage	0.04	0.03	**0.00**	0.04	0.15	30.0	**81.1**	94.1	88.3	87.7
Qwen3.5-4B	one-shot	**2.19**	**0.13**	**0.00**	**1.72**	0.36	20.1	**7.8**	**22.8**	**14.4**	**38.9**
Qwen3.5-4B	two-stage	2.52	0.37	**0.00**	2.32	**0.30**	20.5	5.0	12.7	8.9	7.8
4B-SFT	one-shot	4.31	1.03	0.17	4.31	0.38	68.4	7.8	13.4	9.4	70.6
4B-SFT	two-stage	**0.94**	**0.29**	**0.09**	**0.37**	**0.26**	42.7	**36.1**	**43.7**	**36.7**	**96.1**
4B-RL	one-shot	**0.07**	**0.00**	**0.00**	0.71	**0.01**	25.3	48.9	77.6	73.3	88.3
4B-RL	two-stage	0.08	**0.00**	**0.00**	**0.07**	0.06	27.9	**71.7**	**88.9**	**80.0**	**97.2**
Opus 5	one-shot	0.84	0.51	0.23	**0.01**	**0.00**	36.5	51.1	58.9	51.7	**100.0**
Opus 5	two-stage	**0.09**	**0.16**	**0.12**	0.04	**0.00**	35.0	**62.8**	**81.0**	**63.9**	99.4
Kimi-K3	one-shot	0.78	0.30	0.21	**0.07**	0.01	31.9	52.8	62.0	56.1	**100.0**
Kimi-K3	two-stage	**0.07**	**0.04**	**0.00**	0.09	**0.00**	22.1	**58.3**	**89.8**	**78.3**	98.2

†RL one-shot draws fewer elements; its defect-free rates are achieved on sparse pages, and the fill gate inside LQS discounts them accordingly. Read against Ne as per Table 1 footnote.

need more attempts before a foreign checker accepted its pages, not fewer. Under the same two-stage paradigm the first-pass rate rises from 18.4% on the base model to 87.7% on the trained one, so what was learned holds up against a checker it never trained on. One row needs care: RL's one-shot E2E and First-pass exceed its two-stage values, but on sparse pages ($\mathrm{Ne} = 16.1$); the fill-gated composite reads it correctly (LQS 32.8 against 81.1). The gain therefore is not confined to the fixed, single-instruction setting of our evaluation suite: it carries over intact into a complex engineered pipeline.

6 Conclusion and Limitations

Conclusion. We locate the layout defects of LLM-generated visual code as local planning failure: models recognize their own layout errors and follow given spatial constraints, yet cannot maintain those relations while decoding. This diagnosis yields a two-stage framework separating format-agnostic planning from syntax realization, and a training recipe turning the same geometric signal into an element-level dense reward. A 9B open-source model trained this way raises end-to-end delivery from 25.6% to 85.6%, and its layout ability transfers zero-shot to unseen formats and carries into an agentic pipeline. The distinction that matters is between the scaffold and the training: two-stage inference without training lowers the same model to 17.1%, so what the results establish is that spatial maintenance is a trainable model capability rather than an artifact of the harness.

Limitations. Our reward stops at geometry: aesthetic quality is supervised only indirectly, through the outcome reward's completion terms and the detector's defect feedback, so the gains in visual appeal show how much layout correctness alone contributes to perceived quality—but we try no dedicated aesthetics-level reward, and color harmony, typographic style, and visual hierarchy remain outside the training signal. Resource constraints limited training to 4B and 9B models: in the cross-format experiments they transfer layout ability well but commit frequent syntax errors in unseen formats, a failure we attribute to base-model capacity rather than to the method. One further limit bounds the conclusions: every metric is computed rather than judged by people, so that machine-detected violations track perceived quality is not established by a human study—the most important missing measurement.

AI USE STATEMENT

In this work, we used generative AI tools for the following tasks with required disclosure. *Generating synthetic data*: the SFT trajectories of Section 4.3 were produced by a language model (Claude Opus 5) acting as a teacher under our two-stage procedure, and kept only if they passed the gates described there. *Cleaning and reformatting datasets*: for source corpora that provide only images, we used a language model together with web and slide parsing tools to extract the underlying text content and task description. *Supporting data analysis*: a language model (Fable 5) scores rendered artifacts as a judge, producing the L4 and L3 scores of Appendix A.3 according to the protocols documented there.

We have not used generative AI tools to propose or refine hypotheses, to develop theoretical models or conceptual frameworks, to formulate or prove mathematical claims, or to design or interpret the research. In particular, the research direction, the problem formulation, the design of the diagnostic experiments, the design of the two-stage framework and the training recipe, the experimental protocols, and the interpretation of all reported results are the work of the authors; generative AI contributed no research ideas or methodological decisions. The remaining tasks with required disclosure are not applicable to this work.

Additionally, we used generative AI tools for tasks with recommended disclosure: editing the manuscript for readability, and assisting with the translation of author-written drafts into English.

We have reviewed all AI-assisted work. LLM-generated SFT data was filtered by the gates in Section 4.3; AI-assisted code was executed and inspected by the authors; AI-assisted text was reviewed and revised by the authors; and every number reported in this paper is produced by the released scripts.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

One component of our evaluation suite is a proprietary corpus of clinical medical text. That corpus was fully de-identified before it reached us: it contains no patient names, no identifying dates, no hospital or physician identifiers, and no free-text narrative that could be linked back to an individual, and nothing in the paper reproduces any of its content beyond the aggregate scores reported in the tables. We use it only as source text for a layout task—the model is asked to arrange clinical prose on a page, never to produce or interpret a diagnosis—so the work makes no clinical claim and its outputs are not intended or validated for any medical use. The remaining corpora are public datasets used under the terms of their original releases, and we redistribute none of them.

No human subjects were involved: no human annotators, no human ratings, and no interaction with patients. All assessment reported here is automated, which we note as a limitation in Section 6 rather than as a claim of human-validated quality.

REPRODUCIBILITY STATEMENT

To make the results in this paper checkable, we record here what a reader needs in order to reproduce each reported number, and where it lives.

Data and splits. All evaluation data come from publicly released corpora (WebCode2M, PPTAgent, `pptx_templates`, Slide2Code, Zenodo10K, Design2Code) plus one private clinical corpus; the per-corpus sample sizes of the 450-task evaluation pack are given in Appendix A.3, and the training-set composition — including the verified zero overlap between training decks and the evaluation pack — is in Appendix A.4. Where a source supplies only images, we state in Section 5.2 that the underlying text and task description were extracted with a language model and with web/slide parsing tools.

Model and training. The base model, the four SFT data gates, the number of surviving trajectories, and the RL objective are given in Section 5.2 and Section 4; the full hyperparameters for both stages — optimizer, schedule, batch composition, sequence limits, parallelism, and the GRPO configuration — are listed in Tables 12 and 13.

Reward and metrics. The step reward, the outcome reward, and the advantage combination are specified in Eqs. 1–3, with the outcome reward fully instantiated (decay constant, fill term, gate ramp and coefficients) in Appendix A.4. Every metric is defined in Appendix A.3, which also documents the two adopted external protocols (PPTEval and WebGen-Bench) and the prompts used for the judge stages. All metrics are computed by external scripts applied to rendered or declared geometry, and none of them involves a visual-modality judgment.

REFERENCES

Karthik Ragunath Ananda Kumar and Subrahmanyam Arunachalam. Learning to Present: Inverse Specification Rewards for Agentic Slide Generation. *arXiv preprint arXiv:2603.16839*, 2026.

Tara Boyle. Medical Transcriptions: Sample Medical Transcription Reports. `https://www.kaggle.com/datasets/tboyle10/medicaltranscriptions`, 2018. Data scraped from `https://mtsamples.com`.

Hanqi Chen, Zhongyin Zhao, Ye Chen, Zhujin Liang, and Bingbing Ni. SVGThinker: Instruction-Aligned and Reasoning-Driven Text-to-SVG Generation. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, 2025a.

Jian Chen, Ruiyi Zhang, Yufan Zhou, Rajiv Jain, Zhiqiang Xu, Ryan Rossi, and Changyou Chen. Towards Aligned Layout Generation via Diffusion Model with Aesthetic Constraints. In *International Conference on Learning Representations (ICLR)*, 2024.

Siqi Chen, Xinyu Dong, et al. SVGenius: Benchmarking LLMs in SVG Understanding, Editing and Generation. In *Proceedings of the 33rd ACM International Conference on Multimedia (ACM MM)*, 2025b.

Yang Chen, Yufan Shen, Wenxuan Huang, Sheng Zhou, Qunshu Lin, Xinyu Cai, Zhi Yu, Jiajun Bu, Botian Shi, and Yu Qiao. Learning Only with Images: Visual Reinforcement Learning with Reasoning, Rendering, and Visual Feedback. *arXiv preprint arXiv:2507.20766*, 2025c.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. LayoutGPT: Compositional Visual Planning and Generation with Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 18225–18250, 2023.

Yi Gui, Zhen Li, Yao Wan, Yemin Shi, Hongyu Zhang, Bohua Chen, Yi Su, Dongping Chen, Siyuan Wu, Xing Zhou, et al. WebCode2M: A Real-World Dataset for Code Generation from Webpage Designs. In *Proceedings of the ACM Web Conference (WWW)*, pp. 1834–1845, 2025a.

Yi Gui, Zhen Li, Zhongyi Zhang, Guohao Wang, Tianpeng Lv, Gaoyang Jiang, Yi Liu, Dongping Chen, Yao Wan, Hongyu Zhang, Wenbin Jiang, Xuanhua Shi, and Hai Jin. LaTCoder: Converting Webpage Design to Code with Layout-as-Thought. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2025b.

Daichi Horita, Naoto Inoue, Kotaro Kikuchi, Kota Yamaguchi, and Kiyoharu Aizawa. Retrieval-Augmented Layout Transformer for Content-Aware Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10167–10176, 2023.

Shoma Iwai, Atsuki Osanai, Shunsuke Kitada, and Shinichiro Omachi. Layout-Corrector: Alleviating Layout Sticking Phenomenon in Discrete Diffusion Model. In *European Conference on Computer Vision (ECCV)*, pp. 92–110, 2024.

Dongwei Jiang, Jingyu Zhang, Orion Weller, Nathaniel Weir, Benjamin Van Durme, and Daniel Khashabi. SELF-[IN]CORRECT: LLMs Struggle with Discriminating Self-Generated Responses. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025.

Juyong Jiang, Chenglin Cai, Chansung Park, Jiasi Shen, Sunghun Kim, Jianguo Li, and Yue Wang. WebGen-R1: Incentivizing Large Language Models to Generate Functional and Aesthetic Websites with Reinforcement Learning. *arXiv preprint arXiv:2604.20398*, 2026.

Sha Li, Stefano Petrangeli, Yu Shen, Xiang Chen, and Naren Ramakrishnan. LLMs as Layout Designers: Enhanced Spatial Reasoning for Content-Aware Layout Generation. *arXiv preprint arXiv:2509.16891*, 2025.

Sha Li, Stefano Petrangeli, Yu Shen, and Xiang Chen. From Pixels to Policies: Reinforcing Spatial Reasoning in Language Models for Content-Aware Layout Design. *arXiv preprint arXiv:2602.13912*, 2026a.

Sifan Li et al. GeoSVG-RL: Geometry-Aware Reinforcement Learning for Layout-Constrained Text-to-SVG Diagram Generation. *arXiv preprint arXiv:2605.25447*, 2026b.

Guotao Liang, Zhangcheng Wang, Juncheng Hu, Haitao Zhou, Ziteng Xue, Jing Zhang, Dong Xu, and Qian Yu. Render-in-the-Loop: Vector Graphics Generation via Visual Self-Feedback. *arXiv preprint arXiv:2604.20730*, 2026.

Jiawei Lin, Jiaqi Guo, Shizhao Sun, Weijiang Xu, Ting Liu, Jian-Guang Lou, and Dongmei Zhang. A Parse-Then-Place Approach for Generating Graphic Layouts from Textual Descriptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23622–23631, 2023a.

Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. LayoutPrompter: Awaken the Design Ability of Large Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Shaobo Liu, Feiqiao Mao, Shuaishuai Zhou, Yan Zhan, Weiqi Tan, Zhiqiong Lu, and Zhengping Liang. RefineSVG: Visual Feedback-Driven Reinforcement Learning for Image-to-SVG Generation. In *Proceedings of the 34th ACM International Conference on Multimedia (ACM MM)*, 2026a.

Ziyu Liu et al. Visual-ERM: Reward Modeling for Visual Equivalence. *arXiv preprint arXiv:2603.13224*, 2026b.

Zimu Lu, Yunqiao Yang, Houxing Ren, Haotian Hou, Han Xiao, Ke Wang, Weikang Shi, Aojun Zhou, Mingjie Zhan, and Hongsheng Li. WebGen-Bench: Evaluating LLMs on Generating Interactive and Functional Websites from Scratch. *arXiv preprint arXiv:2505.03733*, 2025.

Gallil Maimon, Ori Yoran, Felix Kreuk, Michael Hassid, Gal Cohen, Pierre Chambon, and Yossi Adi. Self-Execution Simulation Improves Coding Models. *arXiv preprint arXiv:2604.03253*, 2026.

Weijia Mao, Hao Chen, Zhenheng Yang, and Mike Zheng Shou. The Image as Its Own Reward: Reinforcement Learning with Adversarial Reward for Image Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Moonshot AI. Kimi K3: Open Frontier Intelligence. *arXiv preprint arXiv:2607.24653*, 2026.

noxneural. pptx_collection_templates. `https://huggingface.co/datasets/noxneural/pptx_collection_templates`, 2024. Hugging Face dataset of free presentation templates.

Yiming Pan, Chengwei Hu, Xuancheng Huang, Can Huang, Mingming Zhao, Yuean Bi, Xiaohan Zhang, Aohan Zeng, and Linmei Hu. AeSlides: Incentivizing Aesthetic Layout in LLM-Based Slide Generation via Verifiable Rewards. *arXiv preprint arXiv:2604.22840*, 2026.

Zeju Qiu, Weiyang Liu, Haiwen Feng, Zhen Liu, Tim Z. Xiao, Katherine M. Collins, Joshua B. Tenenbaum, Adrian Weller, Michael J. Black, and Bernhard Schölkopf. Can Large Language Models Understand Symbolic Graphics Programs? *arXiv preprint arXiv:2408.08313*, 2024.

Juan A. Rodriguez, Shubham Agarwal, Issam H. Laradji, Pau Rodriguez, David Vazquez, Christopher Pal, and Marco Pedersoli. StarVector: Generating Scalable Vector Graphics Code from Images and Text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025a.

Juan A. Rodriguez, Haotian Zhang, Abhay Puri, Aarash Feizi, Rishav Pramanik, Pascal Wichmann, Arnab Mondal, Mohammad Reza Samsami, Rabiul Awal, Perouz Taslakian, Spandana Gella, Sai Rajeswar, David Vazquez, Christopher Pal, and Marco Pedersoli. Rendering-Aware Reinforcement Learning for Vector Graphics Generation. *arXiv preprint arXiv:2505.20793*, 2025b.

Jaejung Seol, Seojun Kim, and Jaejun Yoo. PosterLlama: Bridging Design Ability of Language Model to Content-Aware Layout Generation. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *arXiv preprint arXiv:2402.03300*, 2024.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. HybridFlow: A Flexible and Efficient RLHF Framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, 2025.

Hengyu Shi, Junhao Su, Tianyang Han, Junfeng Luo, and Jialin Gao. LayoutCoT: Unleashing the Deep Reasoning Potential of Large Language Models for Layout Generation. *arXiv preprint arXiv:2504.10829*, 2025.

Chenglei Si, Yanzhe Zhang, Ryan Li, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. Design2Code: Benchmarking Multimodal Code Generation for Automated Front-End Engineering. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics (NAACL)*, pp. 3956–3974, 2025.

Haoyuan Sun, Jing Wang, Yuxin Song, et al. Power Reinforcement Post-Training of Text-to-Image Models with Super-Linear Advantage Shaping. *arXiv preprint arXiv:2605.10937*, 2026.

Wenxin Tang, Jingyu Xiao, Wenxuan Jiang, Xi Xiao, Yuhang Wang, Xuxin Tang, Qing Li, Yuehe Ma, Junliang Liu, Shisong Tang, and Michael R. Lyu. SlideCoder: Layout-aware RAG-enhanced Hierarchical Slide Generation from Design. *arXiv preprint arXiv:2506.07964*, 2025.

Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. LayoutNUWA: Revealing the Hidden Layout Expertise of Large Language Models. In *International Conference on Learning Representations (ICLR)*, 2024.

Ken Tsui. Self-Correction Bench: Uncovering and Addressing the Self-Correction Blind Spot in Large Language Models. *arXiv preprint arXiv:2507.02778*, 2025.

Yuxuan Wan, Chaozheng Wang, Yi Dong, Wenxuan Wang, Shuqing Li, Yintong Huo, and Michael R. Lyu. Divide-and-Conquer: Generating UI Code from Screenshots. *Proceedings of the ACM on Software Engineering*, 2(FSE):2099–2122, 2025.

Feiyu Wang, Jiayuan Yang, Zhiyuan Zhao, Da Zhang, Bingyu Li, Peng Liu, and Junyu Gao. IntroSVG: Learning from Rendering Feedback for Text-to-SVG Generation via an Introspective Generator-Critic Framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

Jiazhe Wei et al. PosterCopilot: Toward Layout Reasoning and Controllable Editing for Professional Graphic Design. *arXiv preprint arXiv:2512.04082*, 2025.

Haohan Weng, Danqing Huang, Yu Qiao, Zheng Hu, Chin-Yew Lin, Tong Zhang, and C. L. Philip Chen. Desigen: A Pipeline for Controllable Design Template Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12721–12732, 2024.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. Plot2Code: A Comprehensive Benchmark for Evaluating Multi-modal Large Language Models in Code Generation from Scientific Plots. *arXiv preprint arXiv:2405.07990*, 2024.

Ximing Xing, Ziteng Xue, Yandong Guan, Jing Zhang, Dong Xu, and Qian Yu. Reason-SVG: Enhancing Structured Reasoning for Vector Graphics Generation with Reinforcement Learning. *arXiv preprint arXiv:2505.24499*, 2025.

Yajiao Xu et al. CommerceVibe: Learning to Design E-Commerce Creatives as Executable Visual Code via Dual-Feedback Reinforcement Learning. *arXiv preprint arXiv:2608.27893*, 2026a.

Yiming Xu, Jihua Kang, Chunsai Du, Qifan Zhang, Wangqiu Zhou, Yiting Wu, Tianqi Li, and Qi Song. GVR-Coder: A Visual-Feedback Framework for Structured SVG Generation in Complex Document and Meeting Scenarios. *arXiv preprint arXiv:2607.28073*, 2026b.

An Yang et al. Qwen3 Technical Report. *arXiv preprint arXiv:2505.09388*, 2025a.

Cheng Yang, Chufan Shi, et al. ChartMimic: Evaluating LMM's Cross-Modal Reasoning Capability via Chart-to-Code Generation. In *International Conference on Learning Representations (ICLR)*, 2025b.

Tao Yang, Yingmin Luo, Zhongang Qi, Yang Wu, Ying Shan, and Chang Wen Chen. PosterLLaVa: Constructing a Unified Multi-modal Layout Generator with LLM. *arXiv preprint arXiv:2406.02884*, 2024.

Jiayu Yao, Yiwei Wang, Anmeng Zhang, Zhe Sun, Songsong Wang, Lingrui Mei, Yuyao Ge, and Shenghua Liu. Multimodal Reward Hacking in Reinforcement Learning. *arXiv preprint arXiv:2607.09492*, 2026.

Aojie Yuan and Zhiyuan Su. Beyond Language: Format-Agnostic Reasoning Subspaces in Large Language Models. *arXiv preprint arXiv:2605.09496*, 2026.

Sukmin Yun, Haokun Lin, Rusiru Thushara, Mohammad Qazim Bhat, Yongxin Wang, Zutao Jiang, Mingkai Deng, Jinhong Wang, Tianhua Tao, Junbo Li, Haonan Li, Preslav Nakov, Timothy Baldwin, Zhengzhong Liu, Eric P. Xing, Xiaodan Liang, and Zhiqiang Shen. Web2Code: A Large-scale Webpage-to-Code Dataset and Evaluation Framework for Multimodal LLMs. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 37, pp. 112134–112157, 2024.

Junyi Zhang, Jiaqi Guo, Shizhao Sun, Jian-Guang Lou, and Dongmei Zhang. LayoutDiffusion: Improving Graphic Layout Generation by Discrete Diffusion Probabilistic Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Sharon Zhang, R. Kenny Jones, Jiajun Wu, and Maneesh Agrawala. Aggregating LLM-Based Weak Verifiers for Spatial Layout Generation. *arXiv preprint arXiv:2606.05268*, 2026a.

Zhilin Zhang, Xiang Zhang, Jiaqi Wei, Yiwei Xu, and Chenyu You. PosterGen: Aesthetic-Aware Multi-Modal Paper-to-Poster Generation Via Multi-Agent LLMs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 9813–9823, 2026b.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. Zenodo10K. `https://huggingface.co/datasets/Forceless/Zenodo10K`, 2025a. Presentation decks collected from the Zenodo open repository, primarily CC-BY licensed; released with PPTAgent.

Hao Zheng, Xinyan Guan, Hao Kong, Jia Zheng, Hongyu Lin, Yaojie Lu, Ben He, Xianpei Han, and Le Sun. PPTAgent: Generating and Evaluating Presentations Beyond Text-to-Slides. *arXiv preprint arXiv:2501.03936*, 2025b.

Bocheng Zou, Mu Cai, Jianrui Zhang, and Yong Jae Lee. VGBench: Evaluating Large Language Models on Vector Graphics Understanding and Generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3647–3659, 2024.

A Appendix

A.1 Diagnostic Experiments: Locating the Failure

This section reports the controlled experiments behind claims (i)–(iii) of Section 3. We sample about 100 cases from the evaluation suite (its full composition is given in Appendix A.3) and have three models (Opus 5, Kimi-K3 (Moonshot AI, 2026), Qwen3.8-27B) generate one page per case in a single call, under a fixed prompt and on two canvas sizes; each experiment below then varies exactly one factor. These experiments predate the unified instrument of Section A.3 and used an earlier detector version, so we rely on their directions rather than their absolute numbers; every direction below was re-verified under the final instrument.

In this setting, a page generated in a single call carries 4.5 overlapping element pairs on average, and 62% of pages carry at least one. Four candidate causes could explain this: (a) the model cannot *see* its own layout errors; (b) it cannot *follow* spatial constraints; (c) it *forgets* early elements as the context grows; (d) it *runs out of space* as the canvas fills. The experiments below eliminate each in turn, leaving the reading the paper is built on: the model fails to check its immediate neighbors at the moment an element is emitted—local planning failure.

Models recognize their own layout errors (rules out a). Each finished page is handed back to the model that produced it, which must list all overlapping element pairs; the list is checked pair-by-pair against the geometric detector. Agreement is high (Table 4): pair-level F1 lies between 0.65 and 1.00 across the six backend–prompt settings, and no backend falls below 0.86 under the plain prompt. The one degraded setting (Opus 5 on the visually dense prompt, F1 0.65) errs in the harmless direction: it flags many pairs that do not actually overlap (precision 0.51) yet still catches 0.90 of those that do. Seeing the errors is not the bottleneck.

Table 4: Self-judging accuracy: pair-level F1 of the model's own overlap report, scored against the geometric detector, under the plain prompt and the visually dense prompt. The degraded setting fails by over-flagging (precision 0.51) while keeping recall at 0.90.

Backend	Plain prompt	Dense prompt
Opus 5	0.86	0.65
Kimi-K3	0.90	0.98
Qwen3.8-27B	1.00	0.95

Both corpora are pooled in every cell; under the plain prompt Qwen3.8-27B produced no ground-truth overlaps on one corpus, so that cell is read from the other.

Models follow spatial constraints they are given (rules out b). In the same experiment, we added canvas constraints to the prompt to verify whether the model could follow external instructions. The prompt declares four rectangles—about 16% of the canvas—already occupied and off-limits. Without this declaration, 32–43% of elements land inside those rectangles, so they are places the model naturally uses; with it, only 0.0–0.3% do, in every setting, even when pages are made twice as dense (Table 5). Taken together with the previous experiment, this establishes claim (i): the model can recognize a violation after the fact and can obey a constraint handed to it—the two abilities that inference-time engineering schemes rely on. **What it cannot do is maintain, unprompted, the constraints implied by what it has just drawn.** The result also identifies the reliable channel our method exploits: layout decisions survive when moved into the prompt as explicit constraints.

Table 5: Forbidden-region compliance: share of elements landing inside the four declared rectangles, without the declaration (Control) and with it (Declared). Both corpora are pooled in every cell.

	Plain prompt		Dense prompt	
Backend	Control	Declared	Control	Declared
Opus 5	38.5%	0.1%	31.5%	0.2%
Kimi-K3	34.6%	0.0%	32.4%	0.0%
Qwen3.8-27B	42.8%	0.0%	31.8%	0.3%

Errors scale with how much must be tracked at once. Changing only the prompt—from a plain version to one demanding visual hierarchy—roughly doubles the elements per page while models and geometry detector stay fixed. Even after normalizing by the number of element pairs, the overlap rate rises in every backend—by $2.3\times$ and $2.4\times$ for two of them, and notably less for the third (Table 6). Errors grow super-linearly with the number of spatial relations the model must maintain simultaneously—the first sign that the failure lies in maintenance during generation, not in understanding.

Table 6: Density experiment: elements per page (Ne) and overlaps per thousand element pairs, under the plain and the visually dense prompt. Both corpora are pooled in every cell.

	Ne		Overlaps / 1k pairs		
Backend	Plain	Dense	Plain	Dense	Change $\times$
Opus 5	78	112	0.265	0.602	2.3
Kimi-K3	74	105	0.969	1.083	1.1
Qwen3.8-27B	49	89	0.253	0.605	2.4

Errors are local: overlaps involve just-drawn neighbors (rules out c). For every overlapping pair we measure its *memory distance*: the number of elements generated between the two members, with elements numbered in generation order. Table 7 gives the distribution on the evaluation dataset. The median memory distance is 3–4 elements—the figure quoted in Section 1; distances of at most 9 cover 74–86% of all overlapping pairs, and distances over 30 barely occur. **The elements being violated are still well inside the context window.** If the cause were forgetting, overlaps would concentrate at long distances; the opposite holds.

A crowded canvas is not the cause (rules out d). To test whether overlaps come from a crowded page, we freeze the same pages at five stages of generation (Table 8). The page does get fuller: occupied area rises from about 25% to over 80%. If crowding caused the defects, overlaps should increase toward the end. They do not. Overlaps are most frequent at the start, when most of the page is still empty. A lack of space is not the cause. After ruling out (a)–(d), we can name claim (ii): **the failure is local planning, not fading context.** When the model emits an element, it does not check the neighbors already on the page. Fading context would need a longer window; local failure needs the nearby layout to be made explicit as each element is placed, and needs training that teaches the model itself to track the relative positions of neighboring elements. That is what the skeleton (Section 4.1) does before generation, what stepwise rewriting (Section 4.2) does during it, and what the subsequent training (Section 4.3) does.

Errors are never corrected spontaneously, and whole-page repair reproduces the failure. The last experiment asks whether the model corrects its own layout errors after the page is finished. The geometric detector lists every overlapping pair together with its coordinates. The model must then redraw the entire page against that report, for at most four rounds. We collect 79 before/after pairs. Element counts remain comparable across conditions, so a drop in defects is not explained by deleting content (Table 9). Three observations support claim (iii). First, an unnamed error is never revised. The repair procedure's own first draft, produced before any detector report (round 0), stays in the same overlap range as ordinary one-shot generation. Without an external signal that names the violation, the model does not go back. This is the "no backtracking" part of claim (iii). Second, once a violation is named, it can often be removed, but the cost is high. After at most four redraws, Opus 5 reaches zero overlaps per page and the other two backends fall substantially without reaching it, at about two model calls per page (Table 9). A sign test on the paired pages is significant for all three backends. Third, the procedure does not finish the job. After all four rounds, Kimi-K3 and Qwen3.8-27B still carry residual overlaps (0.91 and 1.83 per page). Whole-page redraw asks the model to satisfy every named constraint in a single pass. That is another act of global state maintenance, which claim (ii) already shows the model cannot do. This is why the method later rewrites one local region at a time (Section 4.2). Appendix A.2 later evaluates page-level detect-and-rewrite under the unified instrument, including a control that withholds the detector report (Table 10). Call counts and residual defects under a related feedback protocol are in Table 21. These results establish that post-hoc whole-page repair is costly and unreliable and a misplaced element later serves as a reference, but that causal step is not isolated here. This is why we later rewrite one local region at a time.

Table 7: Memory distance of overlapping pairs—the number of elements generated between the two members of each pair—as a share of all overlaps, over the pages that carry at least one overlap. Both corpora are pooled in every row.

Backend	1–3	4–9	10–29	≥ 30	Median
Opus 5	51%	35%	13%	1%	3
Kimi-K3	48%	25%	22%	5%	4
Qwen3.8-27B	57%	29%	13%	1%	3

Table 8: Canvas fill over generation progress: occupied-area fraction at the first and last of five progress snapshots. Both corpora are pooled in every row.

Backend	Occupied@20%	Occupied@100%
Opus 5	24%	84%
Kimi-K3	27%	84%
Qwen3.8-27B	25%	81%

Table 9: Detect-and-redraw: overlaps per page for plain one-shot generation, the repair procedure's own first draft (round 0), and the final page after up to four repair rounds; average model calls per page. Both corpora are pooled in every row, and element counts (Ne) were verified comparable across conditions.

Backend	One-shot	Round 0	Final	Calls	Ne
Opus 5	5.29	5.26	**0.00**	1.93	125
Kimi-K3	7.77	6.91	0.91	2.00	117
Qwen3.8-27B	4.17	6.13	1.83	2.00	107

Table 10: Training-free intervention grid. Skeleton × feedback loop on five backends, same instrument and same $M = 450$ denominator as Table 1. *null* rows are matched-budget controls in which the detector's report is replaced by uninformative content. Defect families are counts per page; E2E is the end-to-end delivery rate over all M tasks. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Model	Skeleton	Feedback	Ovl↓	Spc↓	Mis↓	Ovf↓	OOB↓	LQS %↑	E2E %↑	Ne
Opus 5	one-shot	—	1.04	1.20	0.35	0.93	0.05	30.2	33.3	60
Opus 5	one-shot	page	0.88	0.89	0.20	0.67	0.02	47.5	46.2	60
Opus 5	one-shot	element	0.50	0.46	0.95	0.96	0.04	34.1	30.4	92
Opus 5	two-stage	—	**0.07**	0.44	0.10	0.12	**0.00**	54.9	54.4	57
Opus 5	two-stage	element	**0.07**	**0.33**	**0.08**	0.10	**0.00**	**63.9**	**54.9**	60
Opus 5	two-stage	element (null)	0.20	0.57	0.21	**0.08**	**0.00**	51.8	42.0	61
Kimi-K3	one-shot	—	3.47	1.93	0.85	0.28	0.04	19.8	16.0	57
Kimi-K3	one-shot	page	1.76	1.57	0.35	0.42	0.01	33.3	24.7	57
Kimi-K3	one-shot	element	1.06	0.32	0.46	0.50	0.08	43.4	27.8	62
Kimi-K3	two-stage	—	0.29	0.55	0.15	**0.01**	**0.00**	49.4	42.9	41
Kimi-K3	two-stage	element	**0.08**	**0.12**	**0.07**	0.02	**0.00**	**71.4**	**51.1**	38
Kimi-K3	two-stage	element (null)	0.28	0.44	0.19	0.03	**0.00**	52.2	39.1	40
Qwen3.8-27B	one-shot	—	2.03	1.09	0.44	0.43	0.05	31.8	34.7	55
Qwen3.8-27B	one-shot	page	1.51	1.05	0.23	0.25	0.02	45.2	41.6	50
Qwen3.8-27B	one-shot	element	0.15	**0.21**	0.22	0.27	**0.00**	53.3	**49.3**	51
Qwen3.8-27B	two-stage	—	0.33	0.57	0.25	**0.03**	**0.00**	39.8	31.3	39
Qwen3.8-27B	two-stage	element	**0.05**	0.29	**0.12**	0.05	**0.00**	**62.4**	48.9	38
Qwen3.8-27B	two-stage	element (null)	0.21	0.63	0.28	0.04	**0.00**	35.4	30.0	39
Qwen3.5-4B	one-shot	—	0.79	0.20	0.01	1.47	0.30	4.3	**32.4**	27
Qwen3.5-4B	one-shot	page	0.44	0.10	**0.00**	1.70	**0.11**	4.4	31.3	25
Qwen3.5-4B	one-shot	element	**0.41**	**0.07**	0.01	1.65	0.22	10.4	29.3	27
Qwen3.5-4B	one-shot	element (null)	0.75	0.22	**0.00**	**1.37**	0.47	6.3	28.4	26
Qwen3.5-4B	two-stage	—	1.88	0.70	0.02	1.66	0.20	12.6	10.9	25
Qwen3.5-4B	two-stage	element	1.13	0.76	0.01	1.84	0.17	**13.9**	13.3	26
Qwen3.5-4B	two-stage	element (null)	1.46	0.96	0.02	1.73	0.26	11.6	10.2	25
Qwen3.5-9B (base)	one-shot	—	0.93	0.21	0.02	2.17	0.46	10.1	**25.6**	32
Qwen3.5-9B (base)	one-shot	page	0.67	1.16	**0.01**	2.43	0.26	9.6	**25.6**	31
Qwen3.5-9B (base)	one-shot	element	0.49	**0.19**	**0.01**	1.96	0.33	11.1	24.9	31
Qwen3.5-9B (base)	one-shot	element (null)	0.82	0.31	0.03	2.34	0.58	10.5	20.4	31
Qwen3.5-9B (base)	two-stage	—	0.67	0.68	0.02	1.75	0.14	14.5	17.1	28
Qwen3.5-9B (base)	two-stage	element	**0.34**	0.66	**0.01**	**1.72**	**0.07**	**19.4**	19.3	29
Qwen3.5-9B (base)	two-stage	element (null)	0.82	0.74	**0.01**	1.77	0.11	13.7	17.3	31

A.2 Training-free Methods and Main Results

To validate the effectiveness of our two-stage reasoning mechanism and the performance gains achieved by the stepwise detect-and-rewrite approach, we designed a series of experiments to verify these improvements and conducted a comprehensive ablation study, thereby demonstrating the efficacy of the strategy.

On the evaluation suite we test a family of training-free inference strategies: *one-shot*, where the model generates the result directly from the task requirements; *two-stage* generation, which first plans a layout skeleton and then realizes it into code; page-level detect-and-rewrite (*page* in Table 10), where the finished page is inspected once by the geometric detector of Section A.3 and the model repairs it against the report; stepwise detect-and-rewrite (*element*), where every fixed number of generation steps the newly produced elements are checked for layout defects and rewritten against the feedback; the combination of the two, i.e. stepwise detect-and-rewrite inside the two-stage framework; and a *null* control of that combination, which keeps every rewrite round but performs no actual inspection or feedback—the model is simply asked to rewrite. Each strategy runs on five backends: Opus 5, Kimi-K3, Qwen3.8-27B, Qwen3.5-9B, and Qwen3.5-4B. Table 10 reports the resulting layout defects under the same instrument and denominator as Table 1.

On the backends whose base capability is sufficient, the table supports three conclusions. First, two-stage generation clearly outperforms one-shot generation across models—LQS rises from 30.2 to 54.9 on Opus 5 and from 19.8 to 49.4 on Kimi-K3, with overlap falling by an order of magnitude—confirming that this inference form alone measurably improves the model's layout ability. Second, detect-and-rewrite is effective, and the finer its granularity the more effective it is: element-level rewriting beats page-level rewriting, which beats one-shot generation ($31.8 \rightarrow 45.2 \rightarrow 53.3$ on Qwen3.8-27B, $19.8 \rightarrow 33.3 \rightarrow 43.4$ on Kimi-K3), and stacked on the two-stage skeleton the element-level loop is the strongest training-free arm on every backend (63.9, 71.4, 62.4). Finally, the comparison between that combined arm and its null control shows that the gain of stepwise detect-and-rewrite is not merely the larger exploration space that repeated redrawing buys—the inspection mechanism itself carries a large share: with the same rewrite rounds but no check signal, LQS drops on every backend ($63.9 \rightarrow 51.8$, $71.4 \rightarrow 52.2$, $62.4 \rightarrow 35.4$). When an explicit external inspection signal is injected, the model's output is visibly better. The improvements on these frontier models suffice to establish the effectiveness of the mechanism.

The two small base models are the exception: on Qwen3.5-9B and Qwen3.5-4B, two-stage generation performs *worse* than one-shot—overlap on the 4B model more than doubles ($0.79 \rightarrow 1.88$), and end-to-end delivery falls from 32.4% to 10.9% on the 4B model and from 25.6% to 17.1% on the 9B model (Table 18). Inspecting the failures traces this to the models' own capability: their instruction following is not strong enough to produce, in the first stage, a skeleton that is both format-compliant and spatially correct—the skeletons they write declare regions that geometrically cannot hold the content assigned to them at rates of 38.8–45.0%, against 0.8–6.0% for the frontier backends (Table 19). This is precisely what our training addresses: it substantially raises this general layout ability of the model itself, which in turn carries the downstream generation further.

A.3 Evaluation Details

Evaluation dataset composition

Here we provide the detailed composition of the evaluation dataset used in the main text in Table 11.

Evaluation metric definitions

Here we provide the detailed definitions of the metrics used in the main text. Every metric below appears in Table 1 or in Appendix A.5, under the abbreviation given in bold. All are computed per page and averaged over scored pages; the denominator is pages, never elements, as per-element normalization would reward drawing fewer elements. Let N be the number of elements on a page, U the number of content units in its source material, and M the number of tasks in a format's suite.

Layout compliance. Five defect families, counted per page. **Ovl** (overlap): box pairs intersecting over 6% of the smaller box, containment excluded. **Spc** (spacing): horizontally overlapping pairs

Table 11: Evaluation-set composition: 450 cases, each recast into all four target formats.

Source	Type	Cases
webcode2m	web pages	120
pptagent	document → slides	60
pptx_templates	slide templates	60
slide2code	reference image → slides	60
zenodo10k	academic-talk slides	60
case50	medical reports	50
d2c (Design2Code)	web pages	40
Total		450

with a vertical gap in $(0, 4\,\mathrm{px})$—neither separated nor flush. **Mis** (misalignment): left edges missing a dominant alignment line by 0.5–$4\,\mathrm{px}$. **Ovf** (text overflow): text exceeding its container. **OOB** (out-of-bounds): boxes past the page boundary.

Ne, the element count per page, is also esential for the evaluation. Fewer elements on the canvas naturally result in fewer layout errors, particularly when the elements are sparsely distributed; therefore, this factor must be taken into account when evaluating a model's layout performance.

LQS (Layout Quality Score) is a single page-level verdict: the fraction of pages that are simultaneously defect-free, sufficiently populated ($\mathrm{fill} = \min(1, N/2.5U) \geq 0.8$), and content-complete ($\mathrm{Cov} \geq 0.9$). Being a conjunction, it cannot be raised by drawing less, copying less, or placing badly. Therefore, it is regarded here as a comprehensive reference metric for gauging the quality of model typesetting.

Content fidelity. **Cov**, content coverage, is the fraction of source content units matched in the artifact. It gates scoring rather than ranking methods—defect rates improve monotonically as content is dropped—so pages with $\mathrm{Cov} < 0.9$ are counted but excluded from all averages.

Decl, the declaration-violation rate, checks the skeleton's own arithmetic and therefore exists only for two-stage inference: the fraction of regions declared to hold more text than they geometrically can, i.e. with $\mathrm{slack} = \mathrm{height}/(\text{declared lines} \times \text{font size} \times \text{line height}) < 1$. For one-shot inference it is N/A rather than 0, so the column is not comparable across inference modes.

L4 is the weighted pass rate of a pre-written checklist, adopting the pattern of WebGen-Bench (Lu et al., 2025) with three-valued verdicts ($\mathrm{YES} = 1$, $\mathrm{PARTIAL} = 0.5$, $\mathrm{NO} = 0$) and a vision judge, Fable 5. Items are binary assertions over content, emphasis, and structure, with geometric assertions and degree words excluded by construction. Each checklist is generated from the source material alone, before any artifact—derived from an artifact, it would omit whatever the model failed to draw—and one versioned checklist per task is shared across all models and arms.

L3-C is the Content score of PPTEval (Zheng et al., 2025b), ported here on its five-point rubric with a multimodal judge (Fable 5) that describes the render first and scores the frozen description second.

Aesthetics. The three geometric features are descriptive only, as their correlation with human preference is limited; we set no threshold on them and do not optimize them. **WS**, whitespace, $= 1 - |\bigcup_i b_i|/|C|$ over element boxes b_i and canvas C, using the union so overlap is not double-counted; higher is not better, the median over human-authored pages being ≈ 0.72, so no direction is marked. **Bal**, balance, $= 1 - (\mathrm{BM}_h + \mathrm{BM}_v)/2$, where BM is the area-weighted difference across each midline; 1 is perfectly even. **Align**, the alignment rate, is the fraction of elements on a shared alignment line, clustered along six axes at $1\,\mathrm{px}$ tolerance.

L3-D is the Design score of PPTEval under the same ported protocol as L3-C, judged by the same model (Fable 5) and covering palette, visual elements, and stylistic coherence. The complete judge prompts and scoring rubrics are shown in Appendix A.3.

Delivery. Over the M tasks of a suite, an artifact is *valid* if it renders (HTML), compiles (LaTeX), or executes to a file (`pptx`); it is *scored* if it is valid and clears the Cov gate; it is *clean* if it is scored with every defect family at zero. The headline figure is **E2E**, end-to-end delivery—the probability that one task yields one clean page—which factors into **Completion**, **Gate**, and **DF**, the conditional

defect-free rate:

$$\underbrace{\frac{\text{clean}}{M}}_{\textbf{E2E}} = \underbrace{\frac{\text{valid}}{M}}_{\textbf{Completion}} \times \underbrace{\frac{\text{scored}}{\text{evaluated}}}_{\textbf{Gate}} \times \underbrace{\frac{\text{clean}}{\text{scored}}}_{\textbf{DF}}.$$

L3 JUDGE PROMPTS

L3 describe stage — Content dimension (input: rendered page image)

Please describe the input slide based on the following three dimensions:
1. Information Density
Whether the slide conveys too lengthy or too little information, resulting in a large white space without colors or images.
2. Content Quality
Check if there are any grammatical errors or unclear expressions of textual content.
3. Images and Relevance
Assess the use of visual aids such as images or icons, their presence, and how well they relate to the theme and content of the slides.
Provide an objective and concise description, focusing solely on the specified dimensions.

L3 describe stage — Design dimension (input: rendered page image)

Describe the input slide based on the following dimensions:
1. Visual Consistency
Evaluate if any stylistic choices affect readability, such as overlapping elements or low contrast.
2. Color Scheme
Identify the colors used in the slide and determine if the design is monochromatic (black and white) or colorful (including gray).
3. Use of Visual Elements
Assess the presence of supporting visual elements, such as backgrounds, textures, patterns, or geometric shapes (e.g., rectangles, circles, bold dividers).
Provide an objective and concise description, focusing solely on the specified dimensions.

L3 scoring stage — Content rubric (L3-C; input: frozen description)

You are an unbiased presentation analysis judge responsible for evaluating the quality of slide content. Please carefully review the provided description of the slide, assessing its content, and provide your judgement in a JSON object containing the reason and score. Each score level requires that all evaluation criteria meet the standards of that level.
Scoring Criteria (Five-Point Scale):
1 Point (Poor):
The text on the slides contains significant grammatical errors or is poorly structured, making it difficult to understand.
2 Points (Below Average):
The slides lack a clear focus, the text is awkwardly phrased, and the overall organization is weak, making it hard to engage the audience.
3 Points (Average):
The slide content is clear and complete but lacks visual aids, resulting in insufficient overall appeal.
4 Points (Good):
The slide content is clear and well-developed, but the images have weak relevance to the theme, limiting the effectiveness of the presentation.
5 Points (Excellent):
The slides are well-developed with a clear focus, and the images and text effectively complement each other to convey the information successfully.
Example Output:
{
`"reason": "xx",`
`"score": int`
}
Input:
`{{descr}}`
Please evaluate the slide step by step, ensuring your judgment strictly adheres to the scoring criteria.

L3 scoring stage — Design rubric (L3-D; input: frozen description)

You are an unbiased presentation analysis judge responsible for evaluating the visual appeal of slides. Please carefully review the provided description of the slide, assessing their aesthetics only, and provide your judgment in a JSON object containing the reason and score. Each score level requires that all evaluation criteria meet the standards of that level.
Scoring Criteria (Five-point scale):
1 Point (Poor):
There is a conflict between slide styles, making the content difficult to read.
2 Points (Fair):
The slide uses monotonous colors(black and white), ensuring readability while lacking visual appeal.
3 Points (Average):
The slide employs a basic color scheme; however, it lacks supplementary visual elements such as icons, backgrounds, images, or geometric shapes(like rectangles), making it look plain.
4 Points (Good):
The slide uses a harmonious color scheme and contains some visual elements(like icons, backgrounds, images, or geometric shapes); however, minor flaws may exist in the overall design.
5 Points (Excellent):
The style of the slide is harmonious and engaging, the use of supplementary visual elements like images and geometric shapes enhances the slide's overall visual appeal.
Example Output:

```
{
"reason": "xx",
"score": int
}
```

Input:
{{descr}}
Please evaluate the slide step by step, ensuring your judgment strictly adheres to the scoring criteria.

A.4 Training Details

Training hyperparameters

Here we provide the detailed training protocol in Table 12 and Table 13.

Table 12: SFT hyperparameters. Full-parameter fine-tuning.

Hyperparameter	Value
Base model	Qwen3.5-9B (Yang et al., 2025a) (full-parameter, bf16)
Learning rate	1×10^{-5}, cosine decay, 3% warmup
Global batch size	64 (micro-batch 1 $\times$ grad. accum. 8 $\times$ 8 GPUs)
Epochs	3 (576 steps)
Max sequence length	20,480
Optimizer	AdamW, $\beta = (0.9, 0.95)$, weight decay 0.1, grad. clip 1.0
Parallelism	DeepSpeed ZeRO-3, gradient checkpointing

Table 13: RL hyperparameters (GRPO (Shao et al., 2024), implemented on verl (Sheng et al., 2025)).

Hyperparameter	Value
Initialization	SFT epoch 3 (step 576)
Algorithm	GRPO with token-level dense reward, $\lambda = 0.5$ (Eq. 3); KL and entropy terms off
Learning rate	1×10^{-6}, constant
Batch	64 prompts/step $\times$ 8 rollouts $= 512$ sequences; mini-batch 16
Sampling	temperature 1.0, top-p 1.0
Length limits	prompt $\leq$ 8,192; response $\leq$ 16,384 (skeleton turn $\leq$ 12,288)
Thinking mode	disabled, matching the SFT distribution
Engine / parallelism	vLLM (TP=1) $\times$ 16 workers; FSDP (HSDP=8); 2 nodes, 16 GPUs

TRAINING DATA COMPOSITION

Here we provide the detailed composition of the training data used in the main text in Table 14 and Table 15.

Table 14: SFT data composition. Each case is split into a skeleton sample and a fill sample, so 5,930 cases yield 11,860 training samples. The three *pptx_sparse* rows (1,600 cases) are sparse-layout tasks reverse-engineered from slide decks; their source corpora share names with evaluation corpora, but the decks are a separate split with verified zero overlap against the 450-case evaluation pack.

Source	Type	Cases
webcode2m_train	web pages	2,903
mtsamples	medical reports	1,427
zenodo10k (pptx_sparse)	academic-talk slides, sparse layouts	738
slide2code (pptx_sparse)	slides, sparse layouts	520
pptx_templates (pptx_sparse)	slide templates, sparse layouts	342
Total		5,930

Table 15: RL data composition. Counts are dataset rows.

Source	Rows
train	2,276
val	64
Total	2,340

RL REWARD DETAILS

Step reward. The detector charges six families: the five defect families reported in Section A.3—**Ovl**, **Spc**, **Mis**, **Ovf**, **OOB**—plus **Decl**, a region declared to hold more text than it geometrically can. Decl is reported under content fidelity rather than as a sixth defect column, since it is a property of the skeleton and charging it in both places would count one planning error twice; in training it must cost something at the token that declared it. Table 16 gives the weights w_f of Eq. 1.

The weights are ordered by how expensive a defect is to not have, not by how often it occurs, and the two orders are nearly opposite: weighting by the record counts in Table 16 would make the total an overlap-and-spacing score. Ovf and OOB are the most expensive because the content is unreadable or absent and cannot be repaired by nudging; Ovl is recoverable by moving one element; Mis is cosmetic and its $4\,\mathrm{px}$ threshold is close enough to noise that a large weight would pay the model to chase the detector; Spc is cheapest deliberately, being the highest-count family with most records marginal against its threshold (median gap 3.02 against $4.0\,\mathrm{px}$). Severities $\sigma(v)$ normalize each record against the threshold that fired it, so the weights are comparable across families. Two corrections follow from measurement on the same corpora: `overlap` and `spacing` records name two elements and are charged to both, so $\rho_f = 0.5$ for them and 1 elsewhere; records whose geometry came from text-width estimation rather than measurement are discounted by half rather than dropped. The floor is $r_{\min} = -1.0$; over 200 products no element reaches it, with per-element penalty median -0.013 and minimum -0.546.

Outcome reward. Eq. 2 instantiates as

$$R = \min\Big(1, \tfrac{c}{0.9}\Big)\Big(0.4\,c + 0.4\,e^{-D/6} + 0.2\min\big(1, \tfrac{N}{1.5\,U}\big)\Big), \tag{4}$$

with c the content coverage, D the defect count excluding OOB, N the element count, and U the source's content-unit count. Instantiated against Eq. 2, the decay constant is $\kappa = 6$, the fill term is $\lambda_{\text{fill}}\,\text{fill}$ with $\text{fill} = \min(1, N/(K\,U))$ and $K = 1.5$, and the gate ramp is $\min(1, c/0.9)$; the coefficients are $(\lambda_c, \lambda_e, \lambda_{\text{fill}}) = (0.4, 0.4, 0.2)$. The weights are calibrated on the offline products of Section A.1, grouped by task so that within-group variance—the only quantity GRPO sees—could be measured: $\kappa = 6$ places the median page at 0.72 and a 10-defect page at 0.19, and $K = 1.5$ saturates pages that are both clean and content-complete (their N/U has $p_{05} = 1.50$) while still

Table 16: Step-reward weights. Records are detector counts over the offline corpora of Section A.1 (10,599 products).

Family	Column	w_f	Records
`overflow`	Ovf	-0.40	5,253
`oob`	OOB	-0.30	5,120
`overlap`	Ovl	-0.25	46,165
`fit`	Decl	-0.20	289
`misalign`	Mis	-0.15	9,057
`spacing`	Spc	-0.10	37,741

separating a truncated one. The coverage gate is a ramp rather than the hard $\mathrm{Cov} \geq 0.9$ cut used for reporting, because a hard cut sends 15.2% of rollout groups to identical scores, and a group with no score variance is one GRPO learns nothing from.

Neither term is safe alone. Deleting half the elements from 40 real pages, a defect-only score rose on 19 of them while coverage collapsed from 0.92 to 0.28; with the gate and fill term in place the same attack succeeds on 1 of 40, and that page is a genuine improvement. We train with $\lambda = 0.5$ in Eq. 3, inside a measured safe range of 0.3–1.0. This range is not where the reward is studied but where it is used: the value sits at the middle of the interval rather than at either end, so that neither the dense term nor the outcome term dominates a combined advantage, and no result in this paper turns on a boundary choice. The $\lambda = 0$ limit of the same objective is the outcome-only arm of Section 5.4, which is what makes that comparison a within-recipe control: the two arms share data, base model, advantage normalization and outcome term, and differ only in whether the dense signal is present.

A.5 Main Results across All Metrics

Table 1 reports a subset chosen for readability. The four tables below give every metric of Appendix A.3 on the same twelve base arms plus the two trained 4B arms, grouped by family: layout compliance (Table 17), delivery (Table 18), content fidelity (Table 19), and aesthetics (Table 20). Unless stated otherwise, geometric columns are computed on each arm's own set of scored pages, whose size is given in Table 18.

Table 17: Layout compliance. Defect families per page, the composite page score, and the element-count guardrail. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Model	Inference	Ovl↓	Spc↓	Mis↓	Ovf↓	OOB↓	LQS %↑	Ne
Opus 5	one-shot	1.04	1.20	0.35	0.93	0.05	30.2	60
Opus 5	two-stage	**0.07**	**0.44**	**0.10**	**0.12**	**0.00**	**54.9**	57
Kimi-K3	one-shot	3.47	1.93	0.85	0.28	0.04	19.8	57
Kimi-K3	two-stage	**0.29**	**0.55**	**0.15**	**0.01**	**0.00**	**49.4**	41
Qwen3.8-27B	one-shot	2.03	1.09	0.44	0.43	0.05	31.8	54
Qwen3.8-27B	two-stage	**0.33**	**0.57**	**0.25**	**0.03**	**0.00**	**39.8**	39
Qwen3.5-4B	one-shot	**0.79**	**0.20**	**0.01**	**1.47**	0.30	4.3	27
Qwen3.5-4B	two-stage	1.88	0.70	0.02	1.66	**0.20**	**13.8**	25
Qwen3.5-9B (base)	one-shot	0.93	**0.21**	**0.02**	2.17	0.46	10.1	32
Qwen3.5-9B (base)	two-stage	**0.67**	0.68	**0.02**	**1.75**	**0.14**	**14.5**	28
9B-SFT (ep3)	two-stage	0.34	0.49	0.15	0.45	0.03	41.4	63
9B-RL (GRPO)	two-stage	**0.01**	**0.03**	**0.00**	**0.06**	**0.00**	**65.5**	48
4B-SFT	two-stage	1.84	1.03	0.18	0.65	0.08	38.5	69
4B-RL	two-stage	**0.02**	**0.02**	**0.01**	**0.05**	**0.00**	**63.4**	44

Table 18: Delivery. Counts are pages; rates are percentages. E2E = completion × gate × DF, with $M = 450$ fixed for every arm. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

		Counts			Rates %			
Model	Inference	Valid	Scored	Clean	Completion↑	Gate↑	DF↑	E2E↑
Opus 5	one-shot	444	374	150	**98.7**	84.2	40.1	33.3
Opus 5	two-stage	441	388	245	98.0	**87.2**	**63.1**	**54.4**
Kimi-K3	one-shot	353	303	72	78.4	**85.8**	23.8	16.0
Kimi-K3	two-stage	368	312	193	**81.8**	84.8	**61.9**	**42.9**
Qwen3.8-27B	one-shot	416	362	156	**92.4**	87.0	43.1	**34.7**
Qwen3.8-27B	two-stage	325	289	141	72.2	**88.9**	**48.8**	31.3
Qwen3.5-4B	one-shot	448	348	146	**99.6**	**77.5**	**42.0**	**32.4**
Qwen3.5-4B	two-stage	383	246	49	85.1	64.2	19.9	10.9
Qwen3.5-9B (base)	one-shot	449	368	115	**99.8**	**81.8**	**31.2**	**25.6**
Qwen3.5-9B (base)	two-stage	407	296	77	90.4	72.7	26.0	17.1
9B-SFT (ep3)	two-stage	437	370	169	97.1	84.7	45.7	37.6
9B-RL (GRPO)	two-stage	441	414	385	**98.0**	**93.9**	**93.0**	**85.6**
4B-SFT	two-stage	434	356	115	96.4	82.0	32.3	25.6
4B-RL	two-stage	440	424	**396**	97.8	**96.4**	**93.4**	**88.0**

Table 19: Content fidelity. Cov and Decl are geometric; L4 and L3-C come from the adopted judge protocols (Fable 5), computed on the common intersection of tasks all compared arms completed. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Model	Inference	Cov↑	Decl %↓	L4↑	L3-C*
Opus 5	one-shot	0.989	N/A	**0.902**	2.62
Opus 5	two-stage	**0.991**	6.0	**0.902**	2.10
Kimi-K3	one-shot	**0.992**	N/A	**0.906**	2.69
Kimi-K3	two-stage	0.991	**0.9**	0.905	2.07
Qwen3.8-27B	one-shot	0.989	N/A	0.874	**2.69**
Qwen3.8-27B	two-stage	**0.994**	0.8	**0.875**	2.17
Qwen3.5-4B	one-shot	0.986	N/A	**0.597**	1.55
Qwen3.5-4B	two-stage	**0.988**	45.0	0.572	1.34
Qwen3.5-9B (base)	one-shot	0.986	N/A	0.660	1.59
Qwen3.5-9B (base)	two-stage	**0.988**	38.8	**0.707**	1.52
9B-SFT (ep3)	two-stage	0.988	7.4	0.853	1.66
9B-RL (GRPO)	two-stage	**0.995**	**5.4**	**0.871**	**1.79**
4B-SFT	two-stage	0.987	8.13	0.819	1.55
4B-RL	two-stage	**0.996**	**1.24**	**0.856**	**1.69**

Table 20: Aesthetics. WS, Bal, and Align are descriptive geometric features with no threshold; L3-D is the PPTEval Design score, judged by Fable 5 on the same common intersection of tasks as L4 and L3-C. Ne is repeated as the guardrail for L3-D, which rewards the presence of decoration. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Model	Inference	WS	Bal↑	Align↑	L3-D↑	Ne
Opus 5	one-shot	0.51	0.75	0.976	3.69	60
Opus 5	two-stage	0.46	**0.78**	**0.991**	**3.83**	57
Kimi-K3	one-shot	0.44	**0.80**	0.957	**3.86**	57
Kimi-K3	two-stage	0.46	0.78	**0.983**	3.79	41
Qwen3.8-27B	one-shot	0.47	0.76	0.971	3.69	54
Qwen3.8-27B	two-stage	0.39	**0.78**	**0.986**	**3.79**	39
Qwen3.5-4B	one-shot	0.78	0.45	0.902	**1.31**	27
Qwen3.5-4B	two-stage	0.63	**0.55**	**0.938**	1.17	25
Qwen3.5-9B (base)	one-shot	0.71	0.53	0.922	1.38	32
Qwen3.5-9B (base)	two-stage	0.61	**0.61**	**0.956**	**1.90**	28
9B-SFT (ep3)	two-stage	0.48	**0.78**	**0.993**	3.34	63
9B-RL (GRPO)	two-stage	0.60	0.69	**0.993**	**3.38**	48
4B-SFT	two-stage	0.48	**0.77**	0.992	2.66	69
4B-RL	two-stage	0.60	0.72	**0.993**	**3.17**	44

A.6 Supplementary Ablation Results

Here, we present some experimental results that were not fully included in the main text due to space constraints.

Following the same ablation logic as Section A.2, and to establish that two-stage inference inside the skill is both effective and necessary, we run a supplementary control: on top of one-shot generation, the model is granted one round of reflection and rewriting (+ *FB*), which separates the gain of the two-stage paradigm itself from the gain of simply retrying once. Table 21 reports the full factorial design of training stage $\times$ inference paradigm $\times$ feedback.

The results show that two-stage inference clearly outperforms the alternatives on every model, which establishes that the improvement does not come from spending more inference calls and cost, but from the paradigm itself and from the ability that training internalizes on top of it. Notably, adding a further feedback-and-rewrite round *after* two-stage generation yields no meaningful gain at any training stage, and in places falls below two-stage alone—further evidence that merely appending rewrite rounds does not by itself translate into improvement.

Table 21: Ablation: feedback contribution across training stages. Model × inference × feedback on the agent harness of the *Generalization within Existing Skills* experiment. FB = stepwise detect-and-rewrite added to each paradigm. DF is the final defect-free rate over scored pages; E2E the end-to-end delivery over all 180 tasks. *Residual* is the mean defect count over pages that passed the first inspection (lower = feedback was more effective at clearing defects). LLM calls are normalized to one-shot = 1.00. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Model	Inference	Ovl↓	Ovf↓	Ne	LQS %↑	DF %↑	E2E %↑	First-pass %↑	Calls	Residual↓
base	one-shot	1.89	1.46	21.8	12.8	25.0	19.4	55.6	1.00	0.61
base	one-shot + FB	1.47	**1.27**	21.2	**15.0**	28.7	**20.6**	**57.2**	1.63	**0.13**
base	two-stage	1.73	2.02	18.9	8.3	20.3	14.4	18.4	2.80	2.56
base	two-stage + FB	**0.89**	1.56	20.8	11.1	**30.8**	17.8	20.0	4.01	0.27
SFT	one-shot	3.47	1.54	52.5	16.7	25.4	19.4	75.6	1.00	0.36
SFT	one-shot + FB	2.71	1.36	51.5	15.6	23.8	18.9	74.4	1.44	0.19
SFT	two-stage	**1.01**	0.36	40.0	**42.8**	**49.7**	**42.8**	95.6	2.88	0.05
SFT	two-stage + FB	1.11	**0.20**	40.6	39.4	45.9	40.6	**96.7**	2.89	**0.00**
RL	one-shot	**0.00**	**0.00**	16.1	32.8	**100.0**	**97.2**	98.3	1.00	0.02
RL	one-shot + FB	**0.00**	0.02	17.4	41.1	99.4	95.0	**99.4**	1.01	**0.01**
RL	two-stage	0.04	0.04	30.0	**81.1**	94.1	88.3	87.7	2.71	0.12
RL	two-stage + FB	0.06	0.04	29.1	77.8	93.3	84.4	88.8	2.91	0.09

A.7 Cross-Format Transfer on the 4B Model

The cross-format experiment of Section 5.5 was run on the 9B line, where the configurations differ in more than one respect. The 4B line provides a cleaner control: base, SFT, and RL share the same parameter scale, so only training varies. Table 22 reports the same three unseen formats under the same intent-layer measurement.

The same conclusion holds at this scale. Per-element overlap falls from base through SFT to RL in every format, by 3.5–37.9× against the base model, and Int-E2E rises on all three (15.3 → 28.4 on HTML, 29.6 → 89.6 on LaTeX, 14.0 → 29.8 on PPTX). Because the 4B line varies training alone, the progression belongs to the training recipe. Two observations follow from comparing the two scales. First, training matters more than size: the 4B RL model matches or exceeds the 9B RL model on all three formats (28.4 vs. 20.7, 89.6 vs. 79.8, 29.8 vs. 23.8), while the two base models split the comparison evenly. Second, the LaTeX cell reproduces the signature of Section 5.5 at a different scale—the RL model extracts coordinates from 99.5% of its products and leaves 92.6% of them geometrically clean, yet none compile—so the syntax-versus-layout split is a property of the recipe rather than of one model.

One row needs care. PPTX under SFT is the only regression in intent terms (14.0 → 9.3), but it is not geometric: its defect counts are normalized by an element count that rises 3.1× (9.8 → 29.9), and per element both families improve (0.718 → 0.305 overlap, 0.192 → 0.081 OOB). A clean page requires overlap and OOB to vanish simultaneously, which becomes harder as a page grows denser; the RL row brings both back (31.6%). Read against Ne throughout, as in Table 1.

Table 22: Zero-shot transfer to unseen code formats on the Qwen3.5-4B line (two-stage inference, 450 tasks per format). Ovl and OOB are counts per page recomputed from the coordinates declared in the code; Int-DF and Int-E2E are defined in the text; Align, Cov, and Decl follow the definitions of Appendix A.3, computed on the same declared geometry. Bounds on extractability apply: the HTML SFT and RL rows sit at 48–60%, so their exact values carry selection noise, though not their direction. Within each comparison group—the arms of one model, or the models of one format—the best value in each column is bold; columns without a direction (Ne, WS) are not marked.

Format	Model	Ne	Ovl↓	OOB↓	Int-DF %↑	Int-E2E %↑	Align↑	Cov↑	Decl %↓
HTML	Base	7.7	3.79	1.70	22.0	15.3	0.928	0.577	51.5
HTML	SFT	17.5	1.67	1.21	37.4	15.6	0.934	0.408	10.3
HTML	RL	21.0	**0.28**	**1.02**	**51.4**	**28.4**	**0.986**	**0.603**	**2.8**
LaTeX	Base	4.1	1.12	0.39	58.8	29.6	0.908	0.554	50.9
LaTeX	SFT	15.6	2.94	0.09	64.4	51.8	0.966	0.437	10.3
LaTeX	RL	9.1	**0.71**	**0.03**	**92.6**	**89.6**	**0.983**	**0.575**	**2.6**
PPTX	Base	9.8	7.04	1.88	25.1	14.0	0.914	**0.648**	64.5
PPTX	SFT	29.9	9.11	2.44	13.2	9.3	0.955	0.411	9.3
PPTX	RL	26.0	**1.58**	**1.34**	**31.6**	**29.8**	**0.974**	0.578	**3.0**

A.8 Case Studies

Figure 3 renders four evaluation tasks, one from each corpus family, under four backends—Opus 5 and Qwen3.5-9B before training, after SFT, and after RL—all with two-stage inference.

Here, we present the final rendered results of code generated by different models for the same task. We show the before and after training results of the code generated by the base model, SFT model, and RL model. Results generated by the state-of-the-art model Opus5 are also presented for comparison. The untrained model almost never meets the intended format: even when the code compiles, the layout is typically wrong and far from the task specification. After SFT, the outputs largely follow the required format, but the overall layout can still look unbalanced, and isolated typesetting errors remain. After RL, the pages are layout-compliant and stylistically richer, with almost no remaining typesetting errors. This shows that our method improves code-generation quality: the base model's trained outputs improve substantially and become comparable to those of Opus5.

The same comparison holds at the smaller scale. Figure 4 renders four further evaluation tasks, again one per corpus family, for the Qwen3.5-4B model at the three training stages alongside Opus 5. The progression matches the 9B setting: the untrained 4B model drifts off the intended format almost entirely, SFT restores the structure while leaving occasional typesetting defects, and RL closes the remaining gap.

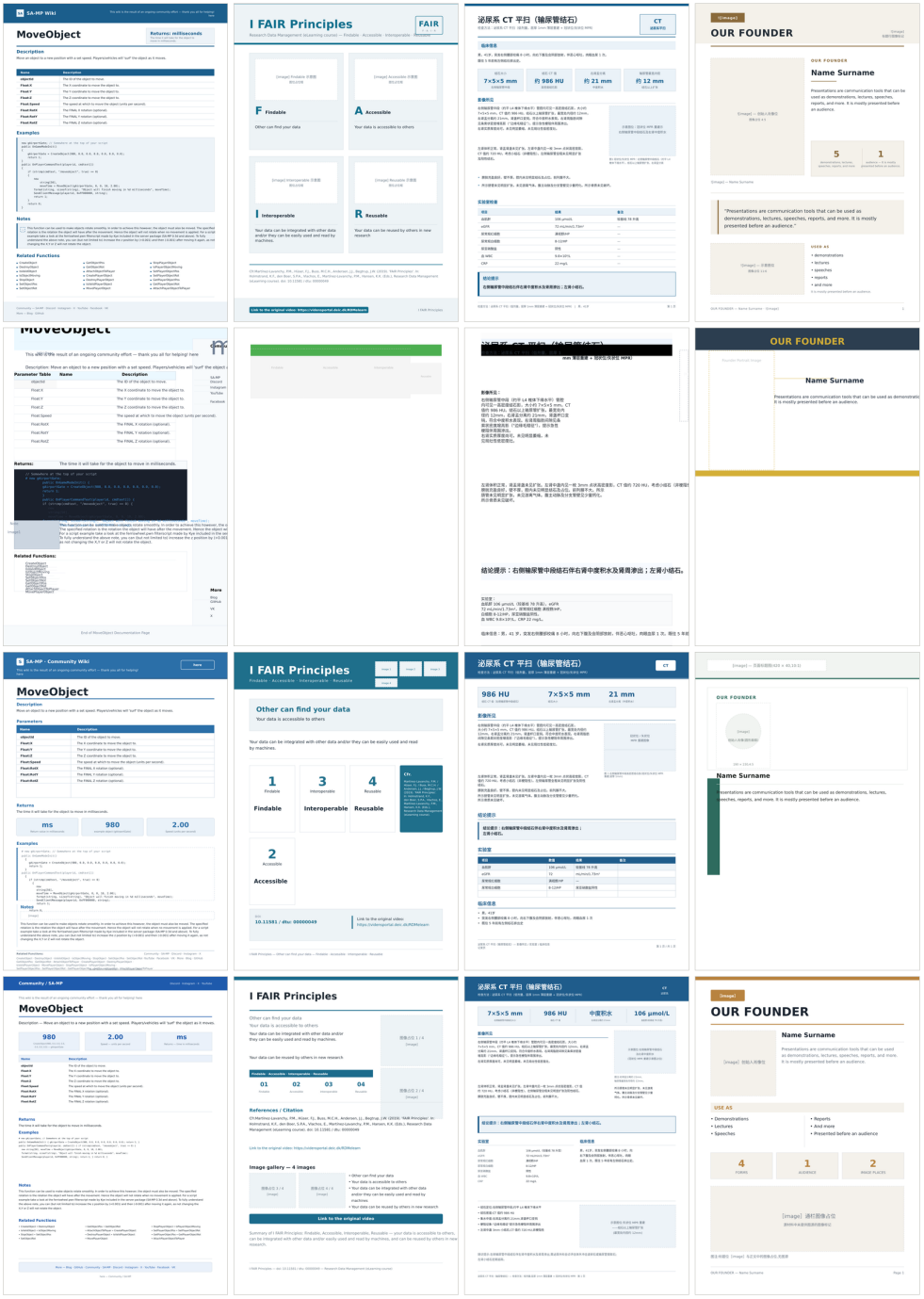


Figure 3: Case studies: four evaluation tasks (columns) rendered by four backends (rows). Tasks, left to right: a technical wiki page (WebCode2M), an academic slide (Slide2Code), a clinical CT report (case50), and a slide template (pptx_templates). The base model clips titles, overlaps blocks, and leaves half the canvas empty; SFT recovers the structure but retains residual defects; RL delivers clean pages comparable to Opus 5.

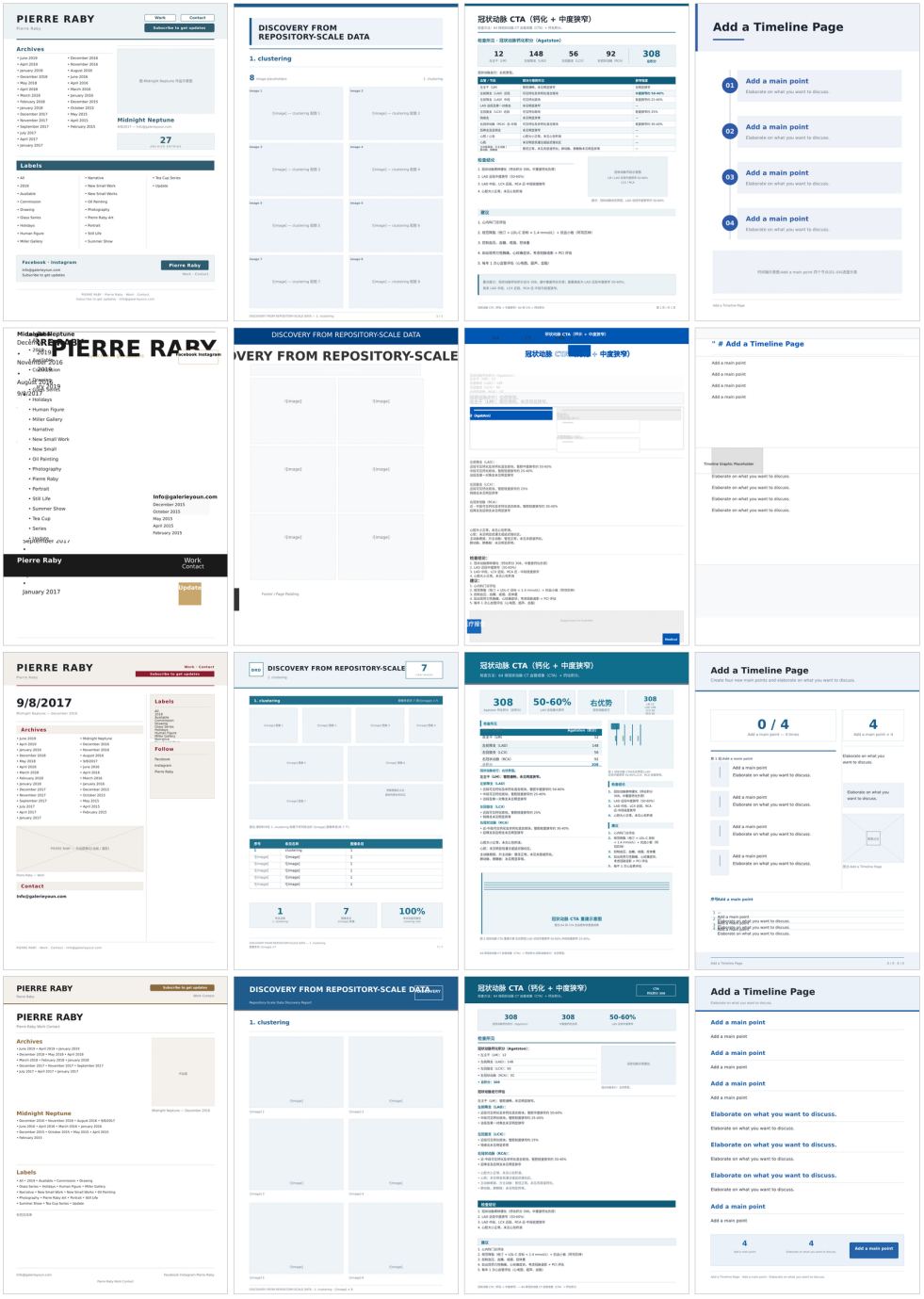


Figure 4: Case studies at the 4B scale: four evaluation tasks (columns) rendered by four backends (rows). Columns, left to right: a technical wiki page (WebCode2M), an academic slide (Slide2Code), a clinical CT report (case50), and a slide template (pptx_templates); the four columns are distinct tasks from those in Figure 3. The 4B base model drifts off the requested format, SFT recovers the structure with residual defects, and RL delivers pages comparable to Opus 5.