RouteDiff: A CONSTRAINT-AWARE DIFFUSION FOR PCB ROUTING WITH GEOMETRY-PRESERVING ROUTE REALIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent learning-based routing methods have focused primarily on VLSI and IC routing, while learned PCB routing remains comparatively underexplored. Printed circuit board (PCB) routing similarly admits multiple valid solutions for a given placement and netlist, with routing geometries that include non-orthogonal segments and are subject to per-net rules such as track width and clearance. We consider PCB routing as sequential conditional generation, where each target-net route is conditioned on the current routing state, target-pad geometry, explicit per-net design rules, and information from the remaining netlist. To model these conditions, we propose ***RouteDiff***, a constraint-aware latent diffusion transformer that integrates spatial routing conditions, scalar design rules, and target-pad geometry through heterogeneous conditioning pathways. We further introduce Geometry-Preserving Route Realization (GPRR), a realization protocol that converts generated routing images into physical PCB traces while limiting modification of the generated geometry. On a public PCB benchmark and an industrial PCB dataset, ***RouteDiff*** achieves the highest F1 to the reference routes and, after GPRR realization, the highest routability and the fewest design-rule violations among the evaluated learned baselines, while responding distinctly to width and clearance interventions. Our evaluations support constraint-aware generation and geometry-preserving realization as complementary components for learned PCB routing.

1 INTRODUCTION

Printed circuit boards (PCBs) are ubiquitous in modern electronic systems, where routing establishes the physical interconnections among placed components under manufacturing and electrical constraints. Yet recent learning-based routing methods, including generative approaches, have focused predominantly on VLSI global routing (Cheng et al., 2022; Du et al., 2023; Shi et al., 2025). Like IC routing, PCB routing seeks to connect individual nets in a placed design and its netlist. For learned PCB routing, the generated route must represent physical trace geometry while accounting explicitly for per-net design rules. PCB routing geometries commonly include non-orthogonal segments, such as 45-degree and any-angle segments (Yan & Wong, 2012; Kohira & Takahashi, 2012), while track widths and clearances vary across signal types and net classes (Lin et al., 2021b; Chang & Chang, 2026). These characteristics motivate explicitly modeling route geometry and per-net constraints during generation.

We consider PCB routing as sequential conditional generation, where the route for each target net is conditioned on the current routing state, target-pad geometry, explicit per-net design rules, including track width and clearance (Lin et al., 2021b), and information from the remaining netlist. Copper occupied by previously routed nets changes the obstacles and available routing space for subsequent nets, making the route of a target net dependent not only on its pads but also on the routed prefix, as in IC and FPGA routing (McMurchie & Ebeling, 1995; Chang et al., 2008). Moreover, multiple routes may satisfy the design rules for the same placement and target net. Consequently, the reference routing in a completed design represents only one valid solution among many. This multimodal solution space motivates modeling a conditional distribution over routing geometry rather than deterministically regressing to a single path. A generative model for PCB routing must there-

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.

fore account for these routing conditions without treating the recorded reference route as the unique solution.

To address these challenges, we propose ***RouteDiff***, a constraint-aware latent diffusion transformer for PCB routing that explicitly incorporates spatial routing conditions, scalar design rules, and target-pad geometry. The spatial conditions encode the current routing context, rule-aware feasibility, and a coarse spatial summary of the remaining netlist. These signals are represented through dedicated spatial, scalar, and token-based pathways, which we refer to as *heterogeneous conditioning*. Spatially aligned routing information is encoded with the routing canvas, track width and clearance modulate the DiT blocks, and variable-length target-pad geometry is injected as coordinate tokens through cross-attention. This conditioning supports layer-aware route generation under the evolving routing state and target-net constraints.

Evaluating routing images with physical metrics such as routability and design-rule violations (DRVs) requires converting them into traces through a route realization procedure, which may include connectivity recovery or search (Du et al., 2023; Shi et al., 2025). These procedures may construct paths absent from the model output, causing predictions of different quality to yield similar routed results and thereby masking differences between generative models. To address this, we introduce Geometry-Preserving Route Realization (GPRR), which realizes generated images as PCB traces while retaining model-specific geometry. GPRR prioritizes geometry already present in the generated image and invokes stronger connectivity recovery only when necessary, limiting the extent to which realization overwrites the prediction. Applied consistently across route-generating models, it provides a common protocol for measuring physical routing quality while retaining generator-specific differences.

Across a public PCB benchmark and a denser industrial PCB dataset, ***RouteDiff*** achieves the highest F1 to the recorded reference routes and, after GPRR realization, the highest routability and the fewest DRVs among the evaluated learned baselines. Compared with an A*-based realization, GPRR retains higher fidelity to the generated routes. Interventions on width and clearance show distinct responses across the scalar and spatial conditioning pathways, while autoregressive rollouts show that generation can continue when previous model outputs define the routing state. As an additional study, we examine WAND, a learned waypoint decoder that converts the learned representation into compact structured traces with routing quality comparable to GPRR.

Our contributions are summarized as follows:

- **Constraint-aware diffusion for PCB routing.** We propose ***RouteDiff***, which explicitly integrates spatial routing conditions, scalar design rules, and target-pad geometry through heterogeneous conditioning.
- **Geometry-preserving route realization.** We introduce GPRR, a common realization protocol that converts generated routing images into physical PCB traces while limiting modification of generator-specific geometry.
- **Evaluation on public and industrial PCBs.** We evaluate reference-route similarity and physical routing quality. ***RouteDiff*** achieves the highest F1 to the recorded reference routes and, after GPRR realization, the highest routability and the fewest DRVs among the evaluated learned baselines, while exhibiting distinct responses to design-rule interventions.

2 Related Work

Routing algorithms and learning-assisted routing. Routing has long been studied across IC, VLSI, and FPGA physical design using search and optimization methods such as maze routing, rip-up and reroute, and negotiated-congestion routing (Lee, 1961; Nair, 1987; McMurchie & Ebeling, 1995; Chang et al., 2008; Liu et al., 2013). For PCB routing, prior work addresses board-level requirements including any-angle geometry, variable track widths and clearances, differential pairs, and hierarchical routing (Kohira & Takahashi, 2012; Lin et al., 2021b;a; Chang & Chang, 2026). Learning-assisted approaches have instead been used for selected routing decisions and intermediate representations, including net ordering, routability, fanout routing, spatial guidance, and Steiner structures (Zhou et al., 2025; Xie et al., 2018; Wang et al., 2022; Yu & Zhang, 2019; Li et al., 2023; Yin et al., 2023; Liu et al., 2021). Our focus is learned route generation, where the model produces a routing representation that can ultimately be realized as a physical route.

Learned route generation and realization. Learned route-generation methods span both PCB and IC routing. For PCB routing, PCB Routing GPT and GPCB predict multi-wire network-flow tokens on a via- and capacity-derived graph and reconstruct physical routes, using a different routing abstraction from our sequential per-net, layer-aware setting (Chen et al., 2025; Wang et al., 2024). For IC and VLSI routing, PRNet, HubRouter, and DSBRouter use conditional GANs, hub-based generation, diffusion models, and diffusion Schrödinger bridges to generate routes or intermediate routing representations (Cheng et al., 2022; Du et al., 2023; Shi et al., 2025). These methods differ in routing abstraction, conditioning information, and output representation. Our formulation focuses on sequential, layer-aware PCB route generation conditioned on the current routing state, target-net geometry, and per-net design rules. We therefore compare against learned spatial routing methods that can be adapted to a common two-layer representation, as summarized in Table 2. Generated routing representations may further require a downstream step to construct or repair connectivity (Cheng et al., 2022; Du et al., 2023). Since this step can alter the generated geometry, we explicitly examine the route realization protocol.

3 Problem Formulation

PCB routing. A PCB contains multiple copper layers, and each *net* specifies a set of component *pads* that must be electrically connected. A physical route consists of copper *traces* on individual layers and *vias* that connect traces across layers. Design rules are specified per net or net class. We model track width and minimum clearance from other copper as explicit per-net design rules. Our experiments focus on signal routing for two-layer PCBs.

Conditional per-net generation. We consider PCB routing under a fixed routing order $\pi = (n_1, \ldots, n_K)$. For a target net n_k, let S_k denote the routing state before n_k is routed, including the board geometry and routes already fixed by the preceding nets under π. Given S_k and the target-net conditions, the goal is to generate routing geometry x_k that connects the target pads. We represent x_k as a layer-aware routing image, with the two copper layers encoded in separate channels.

The conditioning information is organized into three components. The spatial conditions s_k encode the current routing context, rule-aware spatial feasibility, and a coarse spatial summary of the remaining netlist. The vector $\mathbf{r}_k$ contains the scalar design-rule values of the target net, and g_k represents its pad geometry. The model defines the conditional distribution $p_\theta(x_k \mid s_k, \mathbf{r}_k, g_k)$. Because multiple routes may satisfy the same placement and design rules, the route recorded in a completed PCB is treated as one reference solution rather than a unique valid solution.

In the controlled per-net evaluation, S_k is constructed from the reference routes of the nets preceding n_k, providing the same routed prefix to all compared methods. In the autoregressive evaluation, the preceding reference routes are replaced by previously realized model outputs, so that each subsequent routing state is determined by the model's own predictions. Routing-order optimization is outside the scope of this work.

4 Method

4.1 Latent Diffusion Backbone

Figure 1 summarizes the overall ***RouteDiff*** framework. We instantiate the conditional distribution in Section 3 with a latent diffusion model (Ho et al., 2020). Given the routing state and target-net conditions, the model iteratively denoises a latent route representation and decodes it into a layer-aware routing image whose two channels correspond to the two copper layers. The routing conditions enter the denoiser through the heterogeneous conditioning pathways described in Section 4.3. A frozen VAE (Rombach et al., 2022) encodes the routing image into a 64×64 latent representation, where the diffusion process is performed. The denoiser follows a PixArt-Σ-style DiT architecture (Peebles & Xie, 2023; Chen et al., 2024), but uses neither its text encoder nor pretrained DiT weights. The DiT is trained from scratch on routing data with PCB-specific conditioning. Architecture, training details, and capacity and initialization ablations are provided in Appendix A.

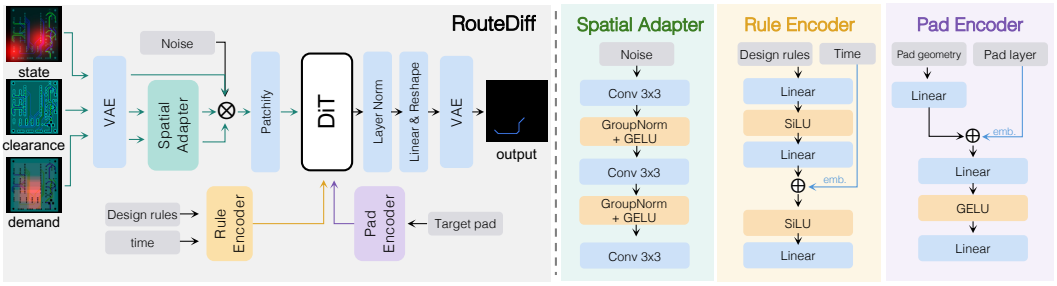


Figure 1: **Overview of *RouteDiff*.** The left panel shows the overall generation pipeline, while the right panels detail the heterogeneous conditioning pathways for spatial routing conditions, scalar design rules, and target-pad geometry.

4.2 Adapting Latent Diffusion to Routing Geometry

Routing images are sparse and composed of thin traces, where small local errors can change connectivity rather than merely appearance. We therefore examine how standard diffusion choices behave in this regime. Table 1 reports matched ablations of these design choices. The plain DiT configuration in (a) is difficult to train reliably under our training schedule, as described in Appendix A.2. Adding logit-normal timestep sampling in (b) improves both training stability and routing performance. However, removing logit-normal sampling from the full recipe in (c) causes only a modest degradation, indicating that its effect depends on the remaining components. Under the full recipe, replacing v-prediction with ϵ-prediction in (d) reduces both reference-route similarity and routability. Removing heterogeneous conditioning in (e) causes a similarly large degradation, showing that explicit routing-state and target-net conditioning contributes beyond the generic DiT backbone. We therefore use v-prediction, logit-normal timestep sampling, and heterogeneous conditioning in the final configuration.

Table 1: **Ablation of the routing-adapted diffusion recipe.** All variants share the same data and optimizer settings and the same validation-based checkpoint selection. Configuration (a) did not complete its schedule. Logit-normal and hetero. cond. denote logit-normal timestep sampling and heterogeneous conditioning, respectively. Results are mean ± std over three sampling seeds, with relative changes Δ from ***RouteDiff*** in parentheses.

	logit-normal	v-pred.	hetero. cond.	F1 (Δ)	Rout. % (Δ)
(a)	–	–	–	0.3864 ±0.0109 (−35.2%)	40.2 ±0.8 (−41.8%)
(b)	✓	–	–	0.5022 ±0.0011 (−15.8%)	59.2 ±1.1 (−14.4%)
(c)	–	✓	✓	0.5844 ±0.0010 (−2.0%)	67.6 ±0.5 (−2.4%)
(d)	✓	–	✓	0.5279 ±0.0075 (−11.5%)	63.8 ±0.5 (−7.7%)
(e)	✓	✓	–	0.5356 ±0.0060 (−10.2%)	62.8 ±0.7 (−9.2%)
RouteDiff	✓	✓	✓	**0.5965** ±0.0020	**69.2** ±0.5

4.3 Routing-Aware Conditioning

As illustrated in Figure 1, ***RouteDiff*** uses dedicated feature-map, embedding, and token-based pathways for spatial routing information, design rules, and target-pad geometry. Appendix Table 19 reports pathway-wise ablations.

Spatial conditions. All board geometry is rasterized into a common coordinate frame while preserving the board aspect ratio, so that spatial distances remain consistent with the physical quantities provided through the scalar pathway. We rasterize the current routing context, board obstacles, and target-pad locations, with Gaussian features emphasizing the target pads (Yin et al., 2023). We additionally construct a clearance-aware field from the distance to occupied regions, offset by the target track width and required clearance. Positive values therefore indicate locations where the target trace has sufficient local space. This exposes the spatial implication of the target-net design rules as a local feasibility signal. The remaining netlist is represented using RUDY (Spindler & Johannes,

2007), which projects net connectivity into a coarse routing-demand field. Although RUDY was originally introduced for routability-driven placement, here it provides the route generator with a spatial estimate of future routing competition without exposing future reference routes. Each spatial condition is encoded by the frozen VAE, and the three resulting latents are concatenated with the noisy routing latent, giving a 16-channel input to the DiT.

Scalar design rules. Track width and clearance are provided as normalized scalar values. Each rule is embedded with a small MLP, and the resulting rule embedding is added to the diffusion timestep embedding. The combined embedding modulates the DiT blocks through adaLN-zero. Both rules are also represented spatially through the clearance-aware field, providing their numerical specification as well as their spatial implications.

Target-pad geometry. Target nets may contain a variable number of pads with different positions and sizes. We represent each pad by its geometry (x, y, s_x, s_y). The geometric values are normalized in the same board coordinate system used for rasterization and projected into a pad token. A layer embedding is added to each token according to the pad's copper-layer type. The resulting variable-length token sequence is injected into the DiT through cross-attention. The rasterized pad locations provide complementary spatial localization without reducing the target net to a fixed start–end pair.

4.4 Route Realization

Image-based routing models require realization before physical evaluation, yet search-based realization may construct geometry absent from the model output. We therefore use GPRR as the common realization protocol and compare its fidelity with an A*-based realization in Section 5.3, while studying learned waypoint decoding as a structured-decoding extension.

Geometry-Preserving Route Realization (GPRR). GPRR adapts a segmentation-to-graph pipeline used in road-network extraction (Máttyus et al., 2017; Van Etten, 2020). It extracts per-layer centerlines from the generated routing image, converts them into octilinear PCB traces, and performs PCB-specific pad snapping, layer-transition inference, and connectivity checking. Layer transitions are inferred from overlapping activity across the two copper layers. When the extracted geometry already connects the target pads, it is retained directly. Additional search is used only to recover missing connections. Applying the same procedure to all route-generating models enables physical routing metrics to be measured while limiting modification of generator-specific geometry. Implementation details are provided in Appendix D.1.

Waypoint Autoregressive Neural Decoder (WAND). GPRR recovers physical traces from the final routing image using a fixed geometric extraction procedure. As an additional study, we investigate whether structured route realization can instead be learned from the representation produced by ***RouteDiff***. WAND autoregressively predicts a sequence of structured routing tokens from an intermediate DiT representation together with the generated routing image, design-rule conditioning, and target-pad tokens. Each token specifies its type, hierarchical x- and y-coordinates, and copper layer, $(\text{type}, x_c, x_f, y_c, y_f, \text{layer})$, allowing lines, branches, and layer transitions to be represented explicitly. Unlike GPRR, WAND directly produces a compact waypoint representation without recovering geometry from a raster skeleton. Section 5.3 evaluates structured waypoint decoding as an additional realization study.

5 Experiments

5.1 Setup

Protocol. We train and evaluate all methods on completed PCB designs. For the public benchmark, we use two-layer PCBWorld D3 boards (Song et al., 2026) spanning all difficulty levels and per-net track width and clearance rules, with 67 test boards and 1,968 target nets. For the industrial evaluation, ***RouteDiff*** and the three strongest public-benchmark baselines by realized F1 are fine-tuned on 227 training boards and evaluated on 15 held-out boards with 2,337 target nets. Ground nets are excluded because they are typically realized as board-wide copper pours rather than routed traces. All methods share the data splits, routing order, target construction, augmentation, and evaluation protocol. Architecture-specific inputs follow the original formulations in Appendix A. Stochastic models are evaluated over three sampling seeds.

Table 2: **Compared methods.** Repre. denotes the representation produced by each learned model. Realization denotes the downstream procedure used by the original method.

Method	Task	Label	Repre.	Realization
GeniusRoute (2019)	Analog IC routing	VAE	Guidance	A*
PRNet (2022)	IC global	cGAN-P	Route	Maze repair
HubRouter (2023)	IC global	cGAN-H / VAE-H / DPM	Hubs	Pin–hub / RSMT
U-Net-A* (2023)	PCB routing	UNet	Guidance	A*
DSBRouter (2025)	IC global	DSB	Route	EG (RSMT)
RouteDiff (ours)	PCB routing	LDM (DiT)	Route	GPRR

Baselines. We compare the guidance-map methods GeniusRoute (Zhu et al., 2019) and U-Net-A* (Yin et al., 2023), realized with PCB A*, and, to include strong learned routers from related domains, adapt PRNet, HubRouter, and DSBRouter (Cheng et al., 2022; Du et al., 2023; Shi et al., 2025) to the common PCB routing target. We retain their original architectures and objectives where compatible, construct inputs from the same PCB state and coordinate frame, and use a common evaluation protocol, replacing domain-specific components, including the original downstream realization. All route-generating methods use the same GPRR procedure. Table 2 summarizes the correspondence between the original methods and our adapted setting.

Metrics. We evaluate both the generated routing geometry and the realized physical routes. *Raw F1* measures layer-aware similarity between the generated image and the reference route, while *realized F1* applies the same metric after route realization. A predicted routing pixel is matched only to reference copper on the same layer within a tolerance of three pixels at 512×512. The binarization threshold is selected separately for each model on the validation split unless otherwise noted. For routing-level evaluation, we additionally report routability and DRVs. Starting from the fully realized routing result, we construct a net-conflict graph from detected DRVs and greedily remove conflicting nets until no violations remain. *Routability* is the fraction of nets retained after this procedure, with disconnected nets counted as removed. *DRVs* report the number of violations before net removal. Both routing-level metrics are specific to this common protocol. Resolution-controlled evaluation, wirelength, and vias are reported in the Appendix.

5.2 Routing Quality on the Public Benchmark

We first ask whether ***RouteDiff*** generates routing geometry that better matches the reference routing, and whether this advantage remains after physical route realization. We use all non-ground nets as evaluation targets. To balance training across boards with different net counts, each epoch draws at most sixty target nets per board without replacement, while evaluation uses every eligible net. Table 3 reports both the generated output and the corresponding physical routing. Among the route-generating methods, ***RouteDiff*** achieves the highest raw F1 against the reference routing. After realization, it also achieves the highest realized F1 and routability with the fewest DRVs among the learned methods. The relative advantage is therefore retained after converting the generated images into physical PCB traces. The corresponding geometric differences are visible in the generated images themselves. The adversarial baselines tend to produce broader routing regions, whereas several diffusion-baseline predictions are more fragmented. ***RouteDiff*** produces sharper, more coherent routing geometry across the two copper layers. Per-net examples are shown in Figure 6 of Appendix B. ***RouteDiff*** remains the strongest model when all route-generating methods are rescored at the lowest baseline resolution of 64×64, as reported in Appendix B. Figure 2 shows qualitative whole-board generated output, using previously realized model outputs as the routing context. We analyze autoregressive behavior in Section 5.5.

5.3 Route Realization and Structured Decoding

We next ask whether routing quality measured after realization remains attributable to the geometry produced by the model rather than to downstream search. To isolate this effect, we apply GPRR and an A*-based realization to the same generated routing images and compare both reference-route similarity and fidelity to the raw model output. Fidelity uses the same layer-aware measure as F1,

Table 3: **Results on the public benchmark.** Raw metrics evaluate generated routing images, while realized metrics are measured after route realization. Stochastic models report mean ± std over three sampling seeds. Routability and DRVs are averaged over boards.

	Raw			Realized		
Model	F1 ↑	Precision ↑	Recall ↑	F1 ↑	Rout. (%) ↑	DRVs ↓
Reference	–	–	–	0.9985	99.3	1.9
A⋆ (uniform)	–	–	–	0.5045	38.4	134.2
VAE	–	–	–	0.4589	36.6	127.6
UNet	–	–	–	0.5394	43.4	123.9
cGAN-P	0.4963	0.508	0.559	0.5563	55.3	111.8
cGAN-H	0.4912	0.511	0.547	0.5541	54.4	107.2
VAE-H	0.0884 ±0.0020	0.071 ±0.002	0.391 ±0.005	0.3030 ±0.0048	25.2 ±1.1	337.9 ±6.9
DPM	0.1632 ±0.0045	0.121 ±0.003	0.381 ±0.006	0.3226 ±0.0062	19.3 ±0.3	311.9 ±0.9
DSB	0.1641 ±0.0011	0.148 ±0.001	0.413 ±0.001	0.2805 ±0.0009	21.6 ±0.7	1104.3 ±12.1
RouteDiff	**0.5965** ±0.0020	**0.600** ±0.002	**0.606** ±0.003	**0.6002** ±0.0024	**69.2** ±0.5	**65.1** ±1.9

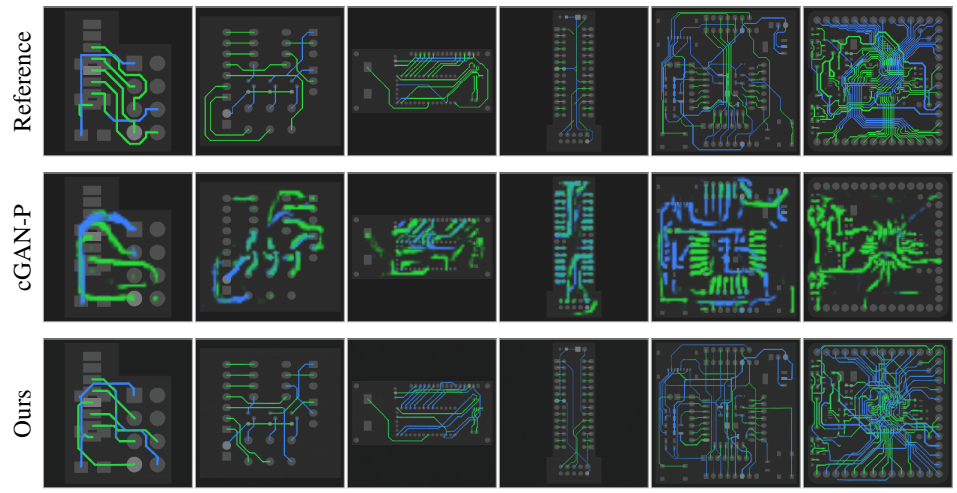


Figure 2: **Qualitative whole-board routing on the public benchmark.** Reference routing and predictions from cGAN-P and ***RouteDiff*** are shown for representative held-out boards. The predictions are generated autoregressively.

but compares the realized traces against the binarized model output instead of the reference routing, so it reports how much of the generated geometry survives realization. Figure 3 shows a clear difference between the two strategies. The A⋆-based realization increases reference-route similarity more strongly for weaker predictions and narrows the differences among the strongest generators. GPRR improves reference-route similarity for all generators, and preserves the raw ordering except for DSB, which drops from fourth to last. More importantly, GPRR consistently retains higher fidelity to the original prediction than the A⋆-based realization. These results show that GPRR realizes physical PCB traces while preserving substantially more of the geometry produced by the generator. Detailed statistics on connectivity recovery and fallback routing are provided in Appendix D.1.

Structured waypoint decoding. We further ask whether ***RouteDiff*** can be realized directly as structured waypoint geometry. We evaluate WAND under both reference-prefix (RP) and autoregressive (AR) contexts. As shown in Figure 3(c), WAND approaches GPRR in routability while producing substantially fewer segments in both settings. Panels (c) and (d) report results over all 1,968 test nets. WAND decodes nets within its 2–8 pad budget, while GPRR realizes the remaining 57 nets. On the 1,911 nets decoded directly by WAND, its routability is within 0.9 points of GPRR while using 3.6× fewer segments. Figure 3(d) further compares encoder and raster-input ablations.

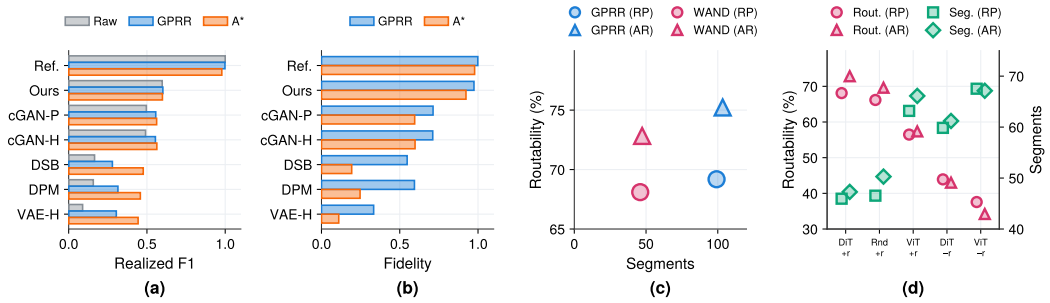


Figure 3: **Route realization across models and structured decoding for *RouteDiff*.** **(a)** Raw and realized reference-route similarity. **(b)** Fidelity to the raw prediction. **(c)** Routability versus segments per net under RP and AR contexts. **(d)** WAND encoder and raster-input ablations, where $+$r and $-$r denote encoders given and denied the generated raster.

The RouteDiff DiT encoder achieves the highest routability among the tested encoders when the raster input is provided, while removing the raster causes a substantially larger degradation. Full results are reported in Appendix D.2.

Table 4: **Results on the industrial PCBs.** Metrics follow Table 3. The reference is evaluated after the same 512×512 rasterize–realize round trip.

	Raw			Realized		
Model	F1 ↑	Precision ↑	Recall ↑	F1 ↑	Rout. (%) ↑	DRVs ↓
Reference	–	–	–	0.9597	84.1	260.0
A (uniform)*	–	–	–	0.6154	46.0	1302.3
UNet	–	–	–	0.6818	50.8	1202.9
cGAN-P	0.6483	0.647	**0.733**	0.7192	37.0	1564.9
cGAN-H	0.6404	0.635	0.727	0.7200	38.0	1565.9
RouteDiff	**0.7071** ±0.0037	**0.809** ±0.001	0.666 ±0.003	**0.7696** ±0.0013	**53.0** ±0.5	**743.8** ±14.0

5.4 Industrial PCB Evaluation

We next test whether the advantage observed on the public benchmark persists on denser production-scale PCBs. For this dataset, learning and evaluation targets are restricted to digital signal nets, while other non-ground nets remain in the predefined routing order and may appear in the reference prefix. We fine-tune ***RouteDiff*** and the three strongest learned baselines by realized F1 on the public benchmark. As shown in Table 4, ***RouteDiff*** retains the highest raw F1, realized F1, and routability among the learned methods, while reducing DRVs by more than half relative to the two adversarial baselines. Thus, the relative advantage observed on the public benchmark persists on the denser industrial boards. Figure 4 illustrates the same trend qualitatively in a dense routing region. The industrial evaluation also exposes a limitation of the 512×512 representation. Even the reference routing reaches only 84.1% routability after the same rasterize–realize round trip, indicating that part of the absolute degradation arises from representation resolution rather than model error alone.

5.5 Autoregressive and Conditioning Analysis

Autoregressive routing. We first evaluate closed-loop autoregressive routing, in which each target net is conditioned on previously realized model outputs rather than the reference prefix, allowing subsequent routing states to depart from the recorded reference trajectory. As shown in Figure 5(a,b), the raw F1 of ***RouteDiff*** decreases under autoregressive context, whereas its routability increases, unlike most baselines, whose routability decreases. Because model-generated prefixes may differ in density from the reference prefix, changes in routability should not be interpreted in isolation as evidence of improved routing decisions. Full autoregressive results are reported in Appendix C.



(a) Reference (b) cGAN-P (c) Ours

Figure 4: **Qualitative comparison on an industrial PCB.** Reference routing and realized predictions are shown for the same PCB. Red callouts magnify the same dense region across panels.

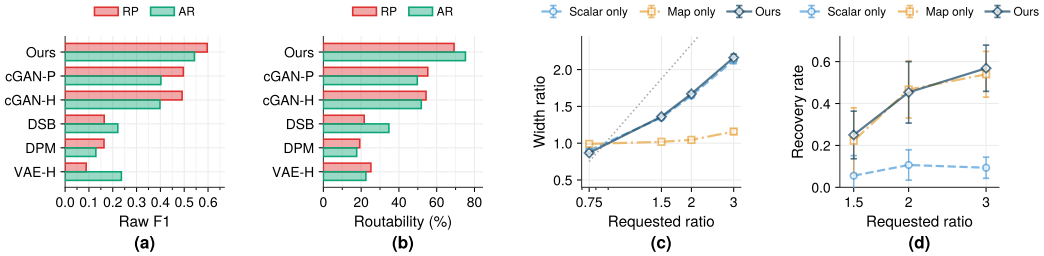


Figure 5: **Behavior under autoregressive context and design-rule interventions.** (a,b) Comparison between reference-prefix (RP) and autoregressive (AR) routing in raw F1 and routability. (c,d) Response to scaled track-width and clearance instructions through the scalar and spatial conditioning pathways. Error bars in (c,d) are board-clustered bootstrap 95% intervals.

Conditioning Analysis. We next examine whether the heterogeneous conditioning pathways contribute to routing quality and whether explicit design-rule inputs induce systematic changes in the generated geometry. Removing any individual conditioning pathway reduces reference-route similarity. The pathways affect different metrics to different degrees, with target-pad tokens contributing more strongly to F1 and the spatial pathway having a larger effect on routability. Detailed pathway-wise ablations are provided in Table 19 of Appendix F. We further probe width and clearance through their scalar and spatial representations. As shown in Figure 5(c), changing the width scalar produces a clear response in generated track width, whereas the spatial field has little effect on thickness. For clearance, the spatial field recovers substantially more binding violations than the scalar alone (Figure 5(d)), suggesting complementary roles in which scalar conditioning primarily specifies track geometry while the spatial field captures local feasibility. Qualitative examples and additional analysis are provided in Appendix F.

6 Conclusion

We proposed ***RouteDiff***, a constraint-aware latent diffusion transformer that conditions route generation on spatial routing information, design rules, and target-pad geometry through heterogeneous conditioning. We introduced GPRR, which converts generated routing images into physical PCB traces while limiting modification of model-specific geometry. Across the public benchmark and industrial PCBs, ***RouteDiff*** achieves the highest reference-route similarity and, after GPRR realization, the highest routability and fewest DRVs among the evaluated learned baselines. Interventions on width and clearance further show complementary responses from the scalar and spatial conditioning, while an additional waypoint-decoding study shows that the learned representation can support compact structured route generation. These findings support explicit routing-condition modeling and geometry-preserving realization as complementary components for learned PCB routing.

THE USE OF LARGE LANGUAGE MODELS

We used generative AI tools as interactive assistants throughout parts of the research and writing process. The core research direction, methodology, experimental questions, and visual presentation were developed by the authors. Generative AI was used to discuss and critique alternative methodological choices, experimental designs, and interpretations of observed results, and to help identify additional analyses or experiments worth considering. It was also used for literature search and summarization, translation, organizing author-provided drafts and notes, and iterative manuscript writing and editing. All methodological decisions, experiment selection, quantitative analyses, figures, and conclusions were determined and verified by the authors. We reviewed all AI-assisted content and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Appendix A provides the model architecture, training and inference configurations, dataset construction procedure, and data-split statistics. Section 5.1 defines the evaluation metrics and protocols used throughout the experiments. Appendix D.1 specifies the GPRR realization procedure and the connectivity-recovery and fallback rules applied consistently across methods, and Appendix D.3 reports its computational cost. Source code is not included with this submission. The public-benchmark experiments can be reproduced from the reported configurations, data construction procedure, and evaluation protocol. The industrial PCB data cannot be redistributed because of data-access restrictions, so exact reproduction of results on this dataset requires access to the corresponding data.

ETHICS STATEMENT

This work uses PCB design data and does not involve human subjects or personal data. The industrial dataset contains proprietary PCB designs and is handled under the applicable data-access restrictions. We do not redistribute the raw industrial board files and report aggregate dataset statistics, quantitative results, and visual examples only for designs approved for publication. The proposed method generates routing geometry and converts it into physical PCB traces for evaluation. The evaluations in this work do not establish manufacturability, electrical correctness, signal integrity, or compliance with all production-level design requirements. Generated routes should therefore undergo standard engineering verification before practical use.

REFERENCES

Hao-Yun Chang and Yao-Wen Chang. Simultaneous power and signal PCB routing considering complex clearance constraints with arbitrary wire widths. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 45(5):2141–2151, 2026. doi: 10.1109/TCAD.2025.3596537.

Yen-Jung Chang, Yu-Ting Lee, and Ting-Chi Wang. NTHU-Route 2.0: A fast and stable global router. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 338–343, 2008. doi: 10.1109/ICCAD.2008.4681595.

Jienan Chen, Jun Tu, Hao Wang, Jie Zheng, Hao Liu, Shenglong Bai, Xiantuo He, and Weikang Qian. GPCB routing: Generative pretrained transformers-based printed circuit board routing method. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 44 (4):1420–1433, 2025. doi: 10.1109/TCAD.2024.3486241.

Junsong Chen, Chongjian Ge, Enze Xie, Yue Wu, Lewei Yao, Xiaozhe Ren, Zhongdao Wang, Ping Luo, Huchuan Lu, and Zhenguo Li. PixArt-σ: Weak-to-strong training of diffusion transformer for 4K text-to-image generation. In *European Conference on Computer Vision (ECCV)*, 2024.

Ruoyu Cheng, Xianglong Lyu, Yang Li, Junjie Ye, Jianye Hao, and Junchi Yan. The policy-gradient placement and generative routing neural networks for chip design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica*, 10(2):112–122, 1973. doi: 10.3138/FM57-6770-U75U-7727.

Xingbo Du, Chonghua Wang, Ruizhe Zhong, and Junchi Yan. HubRouter: Learning global routing via hub generation and pin-hub connection. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. arXiv:2207.12598, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yukihide Kohira and Atsushi Takahashi. An any-angle routing method using quasi-Newton method. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 145–150, 2012. doi: 10.1109/ASPDAC.2012.6164935.

Yao Lai, Yao Mu, and Ping Luo. MaskPlace: Fast chip placement via reinforced visual representation learning. *Advances in Neural Information Processing Systems*, 35:24019–24030, 2022.

C. Y. Lee. An algorithm for path connections and its applications. *IEEE Transactions on Electronic Computers*, EC-10(3):346–365, 1961. doi: 10.1109/TEC.1961.5219222.

Haiyun Li, Jixin Zhang, Ning Xu, and Mingyu Liu. FanoutNet: A neuralized PCB fanout automation method using deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, volume 37, pp. 8554–8561, 2023. doi: 10.1609/aaai.v37i7.26030.

Shih-Ting Lin, Hung-Hsiao Wang, Chia-Yu Kuo, Yolo Chen, and Yih-Lang Li. A complete PCB routing methodology with concurrent hierarchical routing. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1141–1146, 2021a. doi: 10.1109/DAC18074.2021.9586143.

Ting-Chou Lin, Devon J. Merrill, Yen-Yi Wu, Chester Holtz, and Chung-Kuan Cheng. A unified printed circuit board routing algorithm with complicated constraints and differential pairs. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 170–175, 2021b.

Jinwei Liu, Gengjie Chen, and Evangeline F. Y. Young. REST: Constructing rectilinear Steiner minimum tree via reinforcement learning. In *Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1135–1140, 2021. doi: 10.1109/DAC18074.2021.9586209.

Wen-Hao Liu, Wei-Chun Kao, Yih-Lang Li, and Kai-Yuan Chao. NCTU-GR 2.0: Multithreaded collision-aware global routing with bounded-length maze routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(5):709–722, 2013. doi: 10.1109/TCAD.2012.2235124.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv:2211.01095, 2022.

Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. DeepRoadMapper: Extracting road topology from aerial images. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Larry McMurchie and Carl Ebeling. PathFinder: A negotiation-based performance-driven router for FPGAs. In *Proceedings of the ACM International Symposium on Field-Programmable Gate Arrays (FPGA)*, 1995.

Ravi Nair. A simple yet effective technique for global wiring. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 6(2):165–172, 1987. doi: 10.1109/TCAD.1987.1270260.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Urs Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi: 10.1016/S0146-664X(72)80017-0.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations (ICLR)*, 2022.

Liangliang Shi, Shenhui Zhang, Xingbo Du, Nianzu Yang, and Junchi Yan. DSBRouter: End-to-end global routing via diffusion Schrödinger bridge. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 55065–55081, 2025.

Yunqi Shi, Ke Xue, Song Lei, and Chao Qian. Macro placement by wire-mask-guided black-box optimization. *Advances in Neural Information Processing Systems*, 36:6825–6843, 2023.

Hyungseok Song, Junseok Park, Won-Seok Choi, Seohui Bae, Han-Seul Jeong, Youngjoon Park, and Soonyoung Lee. PCBWorld: A benchmark environment for engine-grounded PCB design automation, 2026. URL `https://arxiv.org/abs/2607.05915`.

Peter Spindler and Frank M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *Design, Automation and Test in Europe (DATE)*, 2007.

Adam Van Etten. City-scale road extraction from satellite imagery v2: Road speeds and travel times. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020.

Bowen Wang, Guibao Shen, Dong Li, Jianye Hao, Wulong Liu, Yu Huang, Hongzhong Wu, Yibo Lin, Guangyong Chen, and Pheng Ann Heng. LHNN: Lattice hypergraph neural network for VLSI congestion prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC)*, pp. 1297–1302, 2022. doi: 10.1145/3489517.3530675.

Hao Wang, Jun Tu, Shenglong Bai, Jie Zheng, Weikang Qian, and Jienan Chen. Routing generative pre-trained transformers for printed circuit board. In *Proceedings of the 2nd International Symposium of Electronics Design Automation (ISEDA)*, pp. 160–165, 2024. doi: 10.1109/ISEDA62518.2024.10617906.

Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2018. doi: 10.1145/3240765.3240843.

Tan Yan and Martin D. F. Wong. Correctly modeling the diagonal capacity in escape routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(2):285–293, 2012. doi: 10.1109/TCAD.2011.2169258.

Shiyuan Yin, Min Jin, Gang Chen, Guoliang Gong, Wenyu Mao, and Huaxiang Lu. Unet-Astar: A deep learning-based fast routing algorithm for unified PCB routing. *IEEE Access*, 11:113712–113725, 2023. doi: 10.1109/ACCESS.2023.3323589.

Cunxi Yu and Zhiru Zhang. Painting on placement: Forecasting routing congestion using conditional generative adversarial nets. In *Proceedings of the 56th Annual Design Automation Conference (DAC)*, pp. 1–6, 2019. doi: 10.1145/3316781.3317876.

T. Y. Zhang and Ching Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, 1984. doi: 10.1145/357994.358023.

Zhanwen Zhou, Hankz Hankui Zhuo, Jinghua Zhou, and Wushao Wen. Transformer-based reinforcement learning for net ordering in detailed routing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 9492–9500, 2025. doi: 10.24963/ijcai.2025/1055.

Keren Zhu, Mingjie Liu, Yibo Lin, Biying Xu, Shaolan Li, Xiyuan Tang, Nan Sun, and David Z. Pan. GeniusRoute: A new analog routing paradigm using generative neural network guidance. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2019.

A Reproducibility and Experimental Details

This appendix provides the model configuration, dataset construction, and implementation details used in the experiments. Table 5 summarizes the training and inference settings of ***RouteDiff***, while Tables 6 and 7 report the public and industrial dataset statistics. The training objective is v-prediction (Salimans & Ho, 2022) with logit-normal timestep sampling (Esser et al., 2024), and sampling uses DPM-Solver++ (Lu et al., 2022). Training uses four RTX PRO 6000 GPUs and takes approximately 32 hours for the full 600-epoch schedule.

Table 5: **RouteDiff training and inference configuration.**

Architecture		Training	
Backbone	PixArt-Σ-style DiT	Objective	v-prediction
Layers / hidden dim.	28 / 1152	Timestep sampling	logit-normal $(0, 1)$
Attention heads	16	Optimizer	AdamW
Head dimension	72	Learning rate	10^{-4}
Patch size	2	Weight decay	0.01
Latent resolution	64×64	Warmup	500 steps
Latent channels	16	Effective batch size	32
VAE	frozen SD-VAE, $f = 8$	Precision	bf16
Routing resolution	512×512	Epochs	600
Pad token	(x, y, s_x, s_y)	EMA decay	0.9999
Rule scalars	width, clearance	Augmentation	dihedral (D_4)
Inference		**Industrial fine-tuning**	
Sampler	DPM-Solver	Initialization	public-benchmark checkpoint
Sampling steps	14	Epochs	100
CFG scale	2.5	Learning rate	5×10^{-5}
F1 threshold	$80/255$	Warmup	200 steps

A.1 Dataset Construction and Splits

Routing order and target selection. The routing order is deterministic and identical for every method. Within a board, routable nets are sorted by the area of their pad bounding box in ascending order, with the net name as a tie-break, so that locally confined nets are routed before nets that span the board. A net is routable if it has at least two pads, carries at least one copper segment in the reference design, and is not a ground net. Ground nets are excluded because they are realized as poured planes rather than traces.

Public benchmark. We use two-layer boards from PCBWorld D3. Each training example is constructed by selecting a target net from a completed routing and removing its route while retaining the reference routes of nets preceding it under this order. To reduce imbalance across boards with different net counts, each training epoch draws at most sixty target nets per board without replacement. Validation and test use every eligible target net. Ground nets are excluded. Table 6 summarizes the final split.

Industrial PCBs. The industrial dataset follows the same target-construction procedure. Learning and evaluation targets are restricted to digital signal nets, while routed non-ground nets that are not targets may remain in the conditioning context. The model is initialized from the public-benchmark checkpoint and fine-tuned on the industrial training split. The per-epoch cap is two hundred target nets per board on this dataset. Table 7 reports aggregate statistics only, as the board designs themselves cannot be released.

A.2 Input Conventions and Baseline Adaptation

Board-to-canvas mapping. Each board is mapped to the routing canvas by a uniform letterbox transform: the board bounding box is scaled by a single isotropic factor and centred, leaving a fixed 16-pixel margin at 512×512. Because boards differ in physical size, the resulting pixels-per-millimetre varies by more than an order of magnitude across the corpus.

Table 6: **Public benchmark statistics.** Two-layer boards of PCBWorld D3 (Song et al., 2026). Per-board columns are the mean ± standard deviation over the boards of each split.

			per board			per net
Split	Boards	Nets	Nets	Pads	Components	Pins
train	487	14,471	29.7 ± 32.4	144.7 ± 173.6	42.4 ± 64.8	3.24 ± 4.65
val	64	1,791	28.0 ± 28.2	129.8 ± 134.6	37.4 ± 36.9	3.26 ± 4.27
test	67	1,968	29.4 ± 34.8	126.5 ± 120.9	36.3 ± 41.6	3.03 ± 3.74

Table 7: **Industrial PCB dataset statistics.** Columns follow Table 6, with per-board values given as the mean ± standard deviation over the boards of each split.

			per board			per net
Split	Boards	Nets	Nets	Pads	Components	Pins
train	227	47,761	210.4 ± 190.0	1177.7 ± 1128.3	368.9 ± 318.2	2.97 ± 1.81
val	19	5,955	313.4 ± 190.8	1689.2 ± 1060.1	512.8 ± 295.5	2.85 ± 1.43
test	15	2,337	155.8 ± 141.7	877.7 ± 835.8	299.8 ± 229.4	3.01 ± 2.67

Every raster is drawn above its target resolution and then reduced, so that a change of resolution costs as little geometry as possible. The adapted baselines draw their inputs and targets at 1024×1024 and reduce them to their native grid, and ***RouteDiff*** draws its spatial conditions at $4\times$ and its target at the supersampling factor of the corpus. Enlarging uses the standard image resampling of the vision pipeline, but reducing does not, because averaging a one-pixel trace with its empty neighbours removes it. Copper presence is therefore reduced by taking the maximum over each block, and only the continuous capacity channels of the baselines are averaged. The same rule applies wherever a resolution changes, including the coarse-grid rescoring of Appendix B and the routing context of an autoregressive rollout. All methods share the board frame, the margin, and the exact pad outlines, so the rasterization differs between them only in resolution.

Design-rule normalization. Track width and clearance are supplied to the model in pixels of the board's own canvas rather than in millimetres. A millimetre value has no fixed meaning on the canvas once the scale varies across boards, whereas the pixel value states directly how wide the trace should be drawn, which is what the generator must produce. The same convention is used for the spatial clearance field, so the scalar and spatial routes express the rule in one unit.

Conditioning tensor. The three spatial conditions are the routed-prefix context, the signed clearance-aware feasibility field, and the RUDY demand field of the not-yet-routed nets. Each is rendered as a three-channel image in the board frame and encoded by the frozen VAE. The resulting latents are concatenated channel-wise with the noisy routing latent. Spatial conditions are rendered with $4\times$ supersampling on both datasets, so that sub-pixel geometry appears as grey levels rather than being lost to binarisation. The target routing image is rendered with $2\times$ supersampling on the public benchmark and with $4\times$ on the industrial boards, where the median reference trace is about one pixel wide at 512×512 and a coarser target would quantise it away. The RUDY field is normalised by a single dataset-wide constant so that congestion is comparable across boards. Spatial cost maps of this kind are also used in learned placement (Lai et al., 2022; Shi et al., 2023).

Augmentation. Dihedral (D_4) augmentation is applied jointly to the target image, the spatial conditions, and the target-pad tokens, so that pad coordinates remain consistent with the rasterized geometry. Design-rule scalars are invariant under these transforms and are left unchanged.

Pad geometry. During training, pads are rasterized as rotated rectangles, with circular pads represented as ellipses. At inference and evaluation, we use the exact pad outlines, including oval and rounded-rectangle pads, to avoid rasterization artifacts in design-rule checking.

Routing context. All methods use the same routing order, and each target net is conditioned on the reference routes of preceding target nets. The baseline capacity maps render the vias of the

context nets directly, whereas the context image of ***RouteDiff*** does not and those vias reach the model through the clearance and demand fields instead. On the industrial dataset, where only digital signal nets are evaluation targets, all methods receive routed non-signal nets as context.

Model capacity and native resolution. Table 9 lists the native prediction resolution and the trainable parameter count of each model. We retain the original baseline architectures, so neither resolution nor parameter count is matched. Appendix B controls for prediction resolution, while Table 8 evaluates smaller ***RouteDiff*** variants to assess sensitivity to model capacity.

Table 8: **Capacity and initialisation.** Δ is F1 relative to ***RouteDiff***.

Capacity / init.	Params (M)	F1	Δ	Rout. (%)	DRVs
RouteDiff	**613.1**	**0.5965** $\pm$0.0020	–	**69.2** $\pm$0.5	**65.1** $\pm$1.9
16 layers	358.0	0.5790 $\pm$0.0009	-0.0175	68.7 $\pm$0.5	68.4 $\pm$1.2
8 layers, width 576	48.7	0.5720 $\pm$0.0012	-0.0245	67.1 $\pm$0.8	75.7 $\pm$1.6
initialised from PixArt-Σ	613.1	0.5770 $\pm$0.0032	-0.0195	64.7 $\pm$0.5	80.1 $\pm$1.8

Design-rule checking. Each realized net is expanded beyond its trace half-width by half of its required clearance, and vias by half of the larger of the via diameter and the track width. Two expanded shapes that overlap on the same copper layer therefore violate the clearance rule, since their separation is below the sum of the two half-clearances. A tolerance of one pixel at the evaluation resolution is granted and split between the two parties, so that the rasterize–vectorize round trip of realization does not by itself create violations.

Four classes of obstacle are checked cumulatively: other evaluated nets on the same layer, the pads of other nets, the board outline, and the reference copper of nets that no method was asked to draw. Ground rails are excluded from the last class because they are poured planes rather than routed traces and no method is conditioned on them. A violation is counted once per connected overlap region, and *DRVs* is the total over all four classes.

Routability is computed from the same conflict structure. Pads, the outline, and pre-existing copper cannot be moved, so a net that hits them is removed outright. Conflicts between two evaluated nets are resolved by a greedy vertex cover over the net-conflict graph. A net whose realization does not reach all of its pads is also counted as removed, so that a method cannot appear routable by leaving nets unconnected.

Model selection. Checkpoints are selected for every method on the same fixed validation subset of 500 target nets, and the selected epoch is then evaluated once on the test split. We do not select on the smaller validation probe used to monitor training, because its composition differs from the selection subset and the two disagree on which epoch is best. Binarization thresholds are chosen on the same subset, as described in Appendix E.

Training behaviour of the ablated recipes. The variants of Table 1 share the data, the optimizer and the schedule of the full recipe, and each is evaluated at the epoch chosen on the common validation subset. Configuration (a), which is the backbone with ϵ-prediction and uniform timestep sampling, diverged to a non-finite loss at epoch 225 and did not complete its schedule. Reducing the learning rate did not prevent this, as training diverged at a similar epoch. We therefore apply the same checkpoint-selection procedure as for the other variants to the checkpoints saved before divergence. Configuration (b) differs from it only in the timestep distribution and trained to completion, which is the sense in which logit-normal sampling stabilises this recipe. The remaining variants all completed their schedules.

Implementations. Public benchmark boards are obtained from PCBWorld (Song et al., 2026). We adapt the released implementations of PRNet, HubRouter, and DSBRouter for cGAN-P, cGAN-H, VAE-H, DPM, and DSB. U-Net-A* follows the released architecture and training recipe, reimplemented for our common PCB input representation. Because GeniusRoute does not provide an implementation, we reimplement its predictor from the paper.

Conditioning available to each method. Each adapted baseline receives the inputs of its original formulation, rendered from our board protocol. The generative arms take a target-pin map together with per-direction capacity maps at 64×64. The GeniusRoute-style VAE takes a blurred density of all pads, a blurred density of the target pads, and the context copper occupancy at 64×64. The U-Net-A* predictor takes a Gaussian target-pad emphasis together with the per-layer free space at 256×256. None of these formulations defines an input for per-net design rules or for variable-length pad coordinates, and we do not introduce one, because adding a conditioning pathway would change the method rather than adapt it. ***RouteDiff*** therefore differs from the baselines in the conditioning it consumes as well as in its backbone. Appendix F isolates the contribution of each conditioning pathway within ***RouteDiff*** for this reason. All methods share the board rasterization, the routing order, the routing context, the augmentation, and the evaluation protocol, and every model is trained from scratch on the same public split. The comparison therefore evaluates each model (architecture, objective, and conditioning) under a common data and evaluation protocol. Table 9 lists what each model predicts, how it is trained, and how large it is. The two conditional GANs share a generator and a reconstruction term and differ only in the discriminator: cGAN-P averages the adversarial signal over a two-head discriminator, whereas cGAN-H uses a single PatchGAN.

Table 9: **Adapted baselines.** Native prediction is the representation and resolution each model produces before realization. Parameter counts are trainable parameters of the model actually run, and are not matched across architectures.

Method	Params (M)	Native prediction	Objective	Init.
cGAN-P	11.4	64×64 route	GAN, two-head D + recon.	scratch
cGAN-H	11.4	64×64 route	GAN, PatchGAN D + recon.	scratch
VAE-H	38.2	64×64 route	VAE (KL + recon.)	scratch
DPM	26.5	64×64 route	diffusion	scratch
DSB	43.2	64×64 route	Schrödinger bridge	scratch
U-Net-A*	169.6	256×256 map	segmentation (BCE + Dice)	scratch
RouteDiff	613.1	64×64 latent $\rightarrow 512 \times 512$ route	latent diffusion (v)	scratch DiT

Baseline training budgets. Each baseline follows the optimization recipe of its original formulation, retuned only where the common representation required it. Table 10 lists the budgets actually used. Every arm is trained on the same public split with the same dihedral augmentation, and every checkpoint is selected on the same validation subset of 500 target nets described above.

Table 10: **Baseline training budgets.** Optimizer settings follow each original formulation. All arms share the public split, the dihedral augmentation, and the checkpoint-selection subset.

Method	Optimizer	Epochs	Batch	Learning rate
cGAN-P	Adam $\beta(0.5, 0.999)$	200	64	10^{-4}
cGAN-H	Adam $\beta(0.5, 0.999)$, StepLR	200	64	10^{-4}
VAE-H	Adam, StepLR	200	64	10^{-4}
DPM	Adam	330	64	10^{-4}
DSB	AdamW	1200	96	10^{-5}
VAE	Adam	800	64	10^{-3}
U-Net-A*	Adam, StepLR	100	16	10^{-4}
RouteDiff	AdamW, 500-step warmup	600	32	10^{-4}

A.3 Architecture of the Conditioning Pathways

RouteDiff keeps the PixArt-Σ transformer unchanged and adds three small modules, one per conditioning pathway. All three are initialised so that the model starts from the backbone's initial behaviour and learns to use the PCB-specific signals.

Spatial adapter. The three spatial conditions of Section 4.3, namely the spatial context, the feasibility field, and the routing demand field, are encoded by the frozen VAE into four-channel latents.

The spatial context is passed through unchanged, since it is a rendering of copper geometry and therefore already lies in the distribution the VAE was trained on. The feasibility field and the routing demand field are instead scalar-valued physical quantities rendered as images, so each is refined by a *spatial adapter*: a three-layer residual convolutional block with group normalisation whose output projection is zero-initialised, making the adapter an identity map at the start of training. The refined latents are concatenated channel-wise with the noisy routing latent, giving a sixteen-channel input to the transformer. A learned null tensor replaces the whole spatial block in the unconditional branch of classifier-free guidance.

Rule encoder. Per-net design rules enter as a short vector of scalars together with a presence mask, since a rule may be unspecified for a given net. The *rule encoder* is a two-layer perceptron that maps this vector to a modulation vector, which is added to the diffusion timestep embedding before the backbone's adaptive layer normalisation. Missing rules are replaced by a learned per-slot null feature rather than by a sentinel value, so that "unspecified" is a state the model can represent instead of an arbitrary number it must interpret. The output projection is zero-initialised, so the rule pathway starts as a no-op and the model recovers the unconditional backbone at initialisation.

Pad encoder. The target net is a variable-size set of pads, which the *pad encoder* turns into a token sequence for cross-attention. Each token is the sum of a linear projection of the pad's normalised centre and size, and an embedding of the copper layers the pad occupies. Tokens are emitted at the width of the backbone's text-conditioning stream so that its projection is reused unchanged, which is what allows the existing cross-attention to be repurposed for geometry without architectural surgery. A learned null token represents "no target geometry" in the unconditional branch of classifier-free guidance.

B Additional Benchmark Results

Raw agreement and resolution control. The precision and recall columns of Table 3 decompose raw F1. ***RouteDiff*** maintains both precision and recall around 0.60, whereas VAE-H and the diffusion baselines exhibit substantially higher recall than precision, indicating less selective routing predictions. Because the adapted route-generating baselines predict at 64×64, we additionally rescore all route-generating models on a common 64×64 grid. On this grid a predicted cell must coincide exactly with a reference cell, rather than fall within the three-pixel tolerance used at 512×512. Precision and recall therefore need not move in the same direction as they do at full resolution, and the two adversarial baselines in particular trade recall for precision under the stricter match. ***RouteDiff*** remains the strongest model under this resolution-controlled evaluation. For this comparison, predictions are reduced by 8×8 block-max pooling and the reference is constructed using the same coarse-grid representation. The relative advantage of ***RouteDiff*** therefore does not arise solely from its higher output resolution. The ordering of the remaining models is also preserved, except for the two diffusion baselines, which are within one standard deviation of each other at 512×512 and separate on the coarse grid.

Wirelength and vias. Tables 12 and 13 report additional physical-route statistics after realization. We treat wirelength and via count as descriptive metrics rather than monotonic quality objectives, since shorter routes or fewer vias do not necessarily indicate better agreement with the intended routing geometry.

Qualitative comparison. Figure 6 shows qualitative comparisons on the common 512×512 evaluation canvas. The two GAN-based baselines exhibit broadly similar behavior. They often generate structures that extend between the target pads, but their outputs are frequently diffuse and tend to remain localized around the pad regions rather than forming coherent routes. The VAE baseline produces even blurrier outputs, often with broad activation in the interior of the map rather than a clear trace-like structure. The diffusion-based baselines, DPM and DSB, generate sharper line-like structures with clearer separation between layers, but these structures are often fragmented and less consistently reflect the pad connectivity. In contrast, ***RouteDiff*** produces clearer layer-aware routing geometry that more consistently connects the appropriate pads while avoiding existing routed traces. Although its outputs do not always match the reference exactly, they more often exhibit plausible routing behavior under the given board context.

Table 11: **Raw-output agreement under resolution control.** All route-generating outputs and the reference are reduced to a common 64×64 grid before scoring. The 512×512 scores are in Table 3. The realized columns of that table have no coarse-grid counterpart because realization operates on the evaluation canvas only.

Model	F1 ↑	Precision ↑	Recall ↑
Reference	–	–	–
A (uniform)*	–	–	–
VAE	–	–	–
UNet	–	–	–
cGAN-P	0.4939	0.559	0.507
cGAN-H	0.4882	0.566	0.493
VAE-H	0.0949 ±0.0022	0.077 ±0.002	0.403 ±0.005
DPM	0.1874 ±0.0057	0.164 ±0.005	0.332 ±0.007
DSB	0.1591 ±0.0009	0.143 ±0.001	0.412 ±0.001
RouteDiff	**0.5807** ±0.0010	**0.574** ±0.001	**0.610** ±0.001

Table 12: **Wirelength and via usage on the public benchmark.** WL ratio is the realized wirelength divided by the wirelength of the original board routing. The reference row is the reference routing passed through the same rasterize–realize pipeline, so it is not 1.000 by construction: its distance from 1.000 is the round-trip cost of realization itself, which is small here (0.998) and larger on the finer-pitch industrial boards (1.274, Table 13).

Model	WL (ratio)	Vias / net
Reference	0.998	0.61
A (uniform)*	0.881	0.11
VAE	0.900	0.38
UNet	0.895	0.16
cGAN-P	1.002	0.36
cGAN-H	1.000	0.35
VAE-H	1.059 ±0.031	0.59 ±0.01
DPM	1.157 ±0.019	0.40 ±0.03
DSB	2.390 ±0.083	0.62 ±0.01
RouteDiff	1.020 ±0.002	0.79 ±0.01

Figure 7 shows the same examples on the common 64×64 grid used for resolution-controlled evaluation. The qualitative trends remain consistent at this lower resolution. GAN and VAE baselines remain comparatively diffuse, the diffusion baselines retain sharper local geometry but weaker target-specific structure, and ***RouteDiff*** continues to generate the clearest layer-aware and pad-consistent routing patterns.

C Autoregressive Whole-Board Routing

The main per-net evaluation conditions each target net on the reference routes of preceding nets, and therefore does not expose the model to its own previous predictions. We additionally evaluate full-board autoregressive routing by replacing this reference prefix with the realized outputs produced for preceding nets while keeping the routing order and all other settings unchanged. The evaluation covers all 67 held-out boards and 1,968 target nets. Representative whole-board rollouts are shown in Figure 2.

Replacing the reference prefix with model-generated routes reduces RouteDiff's realized F1 from 0.600 to 0.546, showing the expected loss in reference agreement as prediction errors accumulate. Routability does not decrease accordingly, reaching 75.2% under autoregressive rollout compared with 69.2% under the reference-prefix evaluation.

Table 13: **Wirelength and via usage on the industrial PCBs.** Columns follow Table 12.

Model	WL (ratio)	Vias / net
Reference	1.274	0.60
A (uniform)*	1.091	0.18
UNet	1.114	0.26
cGAN-P	1.336	0.33
cGAN-H	1.306	0.33
RouteDiff	1.201 $\pm$0.002	0.62 $\pm$0.01

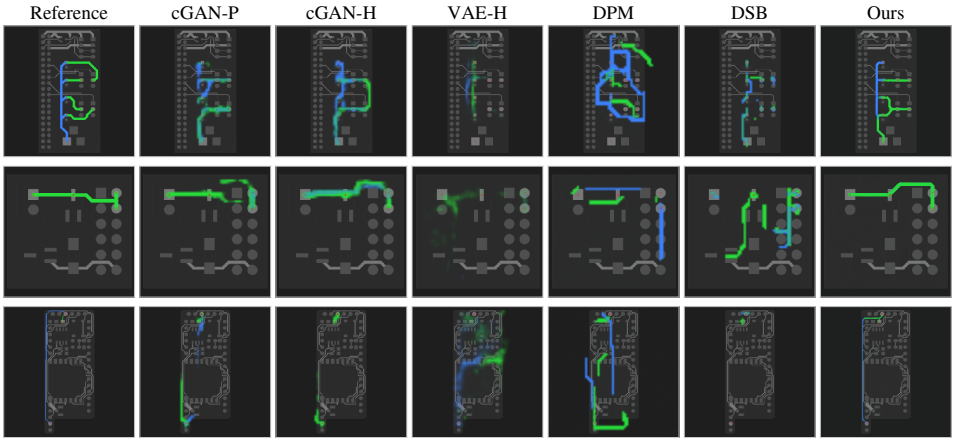


Figure 6: **Per-net qualitative comparison on the public benchmark.** Each row shows one held-out target net and each column a route-generating model. Images are raw outputs before binarization, with green and blue denoting the two copper layers and red circles the target pads.

We do not interpret this numerical increase as evidence that autoregressive routing is easier. The generated prefix is typically sparser than the reference routing, leaving more free space for subsequent nets even as the resulting geometry deviates further from the reference. Despite this distribution shift, RouteDiff retains substantially higher routability and fewer DRVs than the compared baselines.

D Route Realization Details

D.1 GPRR Implementation and Recovery

GPRR processes the two copper-layer predictions independently before recovering inter-layer transitions. Each layer is upsampled by $4\times$, smoothed, and thresholded into a binary mask. Morphological closing along the four octilinear directions reconnects small discontinuities, after which small isolated components are removed.

Concretely, each layer is upsampled by $4\times$, smoothed with a Gaussian of $\sigma = 1.2$ upsampled pixels, and binarized at 0.35. Connected components below 12 upsampled pixels are discarded, and the directional closing uses structuring elements of length 5. Polyline simplification uses a tolerance of $0.03\,\mathrm{mm}$ and branches shorter than $0.3\,\mathrm{mm}$ are pruned. The recovery search penalizes layer changes with a fixed via cost and biases expansion towards the generated copper.

The resulting mask is reduced to a one-pixel centerline using Zhang–Suen thinning (Zhang & Suen, 1984) and interpreted as a graph. Components that do not reach any target pad are discarded. GPRR retains the complete pad-connected skeleton rather than extracting only a shortest path, thereby preserving predicted branches, stubs, and meanders. Branch endpoints are snapped to the target-pad

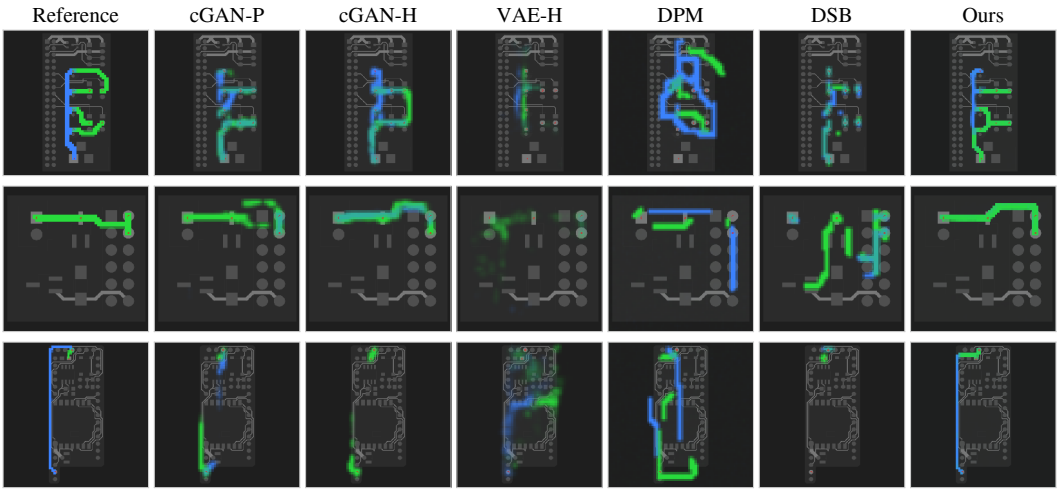


Figure 7: **Resolution-controlled qualitative comparison.** The examples from Figure 6 are shown on the common 64×64 grid used for the resolution-controlled evaluation in Table 11.

Table 14: **Autoregressive whole-board routing.** Eligible target nets are routed sequentially using previously realized model outputs as context rather than the reference prefix. DRVs are counts per board, and bold marks the best model row in each column.

	Raw			Realized		
Model	F1 ↑	Precision ↑	Recall ↑	F1 ↑	Rout. (%) ↑	DRVs ↓
cGAN-P	0.4021	0.429	0.477	0.4764	49.7	124.2
cGAN-H	0.3983	0.412	0.494	0.4722	51.8	128.1
VAE-H	0.1156 ±0.0861	0.114 ±0.116	0.439 ±0.079	0.3103 ±0.0956	22.5 ±10.7	1123.0 ±1132.3
DPM	0.1477 ±0.0140	0.106 ±0.012	0.367 ±0.005	0.3086 ±0.0108	17.7 ±1.1	465.7 ±29.0
DSB	0.2232 ±0.0014	0.280 ±0.001	0.304 ±0.003	0.3971 ±0.0012	34.7 ±0.6	140.0 ±1.8
RouteDiff	**0.5427** ±0.0011	**0.544** ±0.002	**0.556** ±0.001	**0.5462** ±0.0004	**75.2** ±0.1	**55.6** ±1.0

centers, simplified using Ramer–Douglas–Peucker (Ramer, 1972; Douglas & Peucker, 1973), and projected to octilinear segments while keeping their endpoints fixed.

Layer transitions. Vias are inferred from regions in which both copper layers are active. An overlap is accepted as a layer transition only when the centerline of each layer terminates within the region and the overlap is not already covered by a target through-hole pad. This rule avoids interpreting ordinary inter-layer crossings or through-hole pads as vias. On reference routing images from twelve held-out boards, it yields 0.39 inferred vias per net compared with 0.34 in the original designs, whereas accepting every overlapping region yields 0.53.

Table 15 reports the stage at which each realized route becomes connected. For ***RouteDiff***, the extracted geometry is already connected for 74.0% of target nets, while 18.5% require only local bridging and 7.5% are rerouted from scratch. In contrast, the adversarial baselines are directly connected for about one third of their predictions, VAE-H and DPM substantially less often (6.9% and 14.1%), and DSB at 27.4%. The reference routing itself reaches the direct case on 94.5% of nets, showing that part of the residual recovery is introduced by rasterization and extraction rather than by generation alone. Consequently, models that trigger full rerouting more frequently have a larger fraction of their realized score determined by downstream search, complementing the fidelity analysis in Figure 3.

Recovery setting. The connectivity-recovery search is intentionally permissive and does not treat existing copper as a hard obstacle. Violations introduced by this recovery are therefore retained and

counted in the reported DRVs. We use this setting consistently across all route-generating models because an obstacle-aware recovery may fail to return a connected route, which would make the realized comparisons depend on model-specific failure rates. All GPRR extraction and recovery hyperparameters are fixed across route-generating models and are not tuned per model. The model-specific binarization thresholds used for raw-image scoring do not affect the fixed GPRR extraction parameters.

Table 15: **Connectivity recovery stages under GPRR.** Direct routes require no search, bridged routes retain the extracted geometry with local connectivity recovery, and rerouted routes are replaced by full A* search.

Model	Direct (%)	Bridged (%)	Rerouted (%)	Bridges / net
Reference	94.5	5.4	0.1	0.07
cGAN-P	35.9	47.9	16.2	1.16
cGAN-H	36.3	48.0	15.8	1.15
VAE-H	6.9	82.0	11.1	1.80
DPM	14.1	75.7	10.1	1.61
DSB	27.4	55.7	16.9	1.24
RouteDiff	74.0	18.5	7.5	0.56

The mask, thinning, and graph-construction stages follow common segmentation-to-network pipelines used in road extraction (Máttyus et al., 2017; Van Etten, 2020). GPRR adapts this pipeline to PCB routing through layer-aware extraction, pad snapping, via inference, and connectivity recovery that is invoked only for disconnected generated geometry.

D.2 Structured Waypoint Decoding

Representation and decoder. WAND provides a learned realization path from ***RouteDiff*** to explicit waypoint geometry. Rather than encoding the generated raster with a separate network, we reuse the frozen RouteDiff DiT as the representation encoder. The generated routing image is first encoded by the frozen VAE and placed in the routing-latent channels together with the spatial conditioning channels used by ***RouteDiff***. Target-pad tokens and design-rule scalars are provided through the same conditioning pathways as in the diffusion model. We then extract the output of the final, 28th DiT block. This produces 1,024 tokens of dimension 1,152, corresponding to the 32×32 DiT patch grid. The DiT is kept frozen throughout decoder training. We use a small diffusion timestep $t = 50$ while feeding the clean generated-route latent, so that the extracted representation corresponds to a near-denoised routing state.

The decoder memory consists of the DiT tokens together with the target-pad tokens and a design-rule token. DiT tokens are normalized and linearly projected from 1,152 to the decoder dimension of 768, with learned positional embeddings added on the 32×32 grid. Pad and rule tokens are projected into the same feature dimension and concatenated with the DiT tokens. The autoregressive decoder contains six Transformer decoder layers with hidden dimension 768, eight attention heads, a 3,072-dimensional feed-forward layer, pre-normalization, and dropout 0.1. It uses unrestricted causal self-attention and cross-attends to the fixed decoder memory.

Waypoint representation. A route is represented as a sequence of structured vertices, with each vertex encoded by six tokens,

$$(\text{type}, x_c, x_f, y_c, y_f, \text{layer}).$$

The coarse and fine coordinate tokens jointly represent 4,096 normalized coordinate bins through a 64×64 factorization. The vocabulary contains 265 tokens, including five element types, 64 tokens for each coarse and fine coordinate axis, and two layer tokens. Slot-specific masking restricts each position to its valid token subset, preventing syntactically invalid sequences.

A net is represented as a list of chains. Each chain starts from a target pad and terminates when it reaches previously realized copper. The first chain begins with a START element whose pad coordinates are provided to the decoder. Subsequent chains begin with BRANCH elements. LINE represents an in-layer segment, while VIA indicates a layer transition at the current coordinate. An

EOS token terminates the net. Pad coordinates that are known from the target geometry are injected directly and excluded from the prediction loss. The maximum sequence contains 64 route elements, corresponding to 385 tokens including the initial BOS token.

Training. WAND is trained with standard next-token teacher forcing. The objective is cross-entropy over predicted tokens, excluding injected pad coordinates and padding tokens, with no auxiliary losses. We train the decoder for 100 epochs using AdamW with learning rate 3×10^{-4}, weight decay 0.05, batch size 32, and 500 warmup steps. The RouteDiff VAE and DiT remain frozen.

Decoding and search. At inference, WAND generates multiple complete route candidates and reranks them after decoding. We use two candidate root pads and generate four candidates from each root, consisting of one greedy decode and three stochastic decodes at temperature 1.0. This yields eight candidates per net. The selected candidate maximizes

$$s = \overline{\log p} - 2\, N_{\mathrm{rev}} - 10\, \mathbb{I}[\mathrm{unfinished}],$$

where $\overline{\log p}$ is the mean log-probability of predicted tokens, N_{rev} counts turns exceeding 135°, and unfinished sequences receive a fixed penalty.

The generated raster is also used as soft geometric guidance during decoding. For each candidate y-coordinate, we sample the segment from the previous waypoint to the candidate point and penalize distance from the generated copper. This guidance is applied only to the y-coordinate logits because the x-coordinate is generated first. When a chain is attached to existing copper, attachment is allowed only within 64 pixels. All other geometric penalties are disabled in the reported setting.

Test-time scaling. WAND uses best-of-eight decoding rather than beam search. Reranking eight completed candidates instead of keeping the single greedy decode reduces reversing turns per net from 0.71 to 0.16, raises the fraction of sequences that terminate on their own from 0.993 to 1.000, and reduces segments per net from 9.7 to 7.7. The effect saturates at eight candidates. These differences do not appear in routability, which moves by 2.3 points across the three diffusion seeds and therefore cannot separate decoding budgets. Beam search was also less effective, as sequence-level criteria such as termination and reversal count cannot be evaluated reliably on partial routes. We therefore use eight independently decoded candidates with the reranking objective above for all reported WAND results.

What the frozen representation contributes. Across both context regimes and both populations, the frozen ***RouteDiff*** encoder gives the highest routability and the fewest violations among the tested encoders, while a randomly initialised encoder of the same shape matches or slightly exceeds it on realized F1. The learned representation therefore helps the decoder produce routes that survive design-rule checking rather than routes that agree more closely with the reference, and the raster input remains the dominant factor for both.

D.3 Inference Cost

Table 18 reports per-net cost on the public benchmark. Generation is model inference on one GPU, measured for every arm in the same sequential run so that the arms do not contend for the device. Realization applies GPRR and full A* rerouting to the same generated predictions on the same CPU.

GPRR is substantially faster on ***RouteDiff*** outputs than on the weaker generative baselines. Its cost depends on the complexity of the extracted geometry and the amount of connectivity recovery required, with full-board search invoked only when fallback rerouting is necessary.

As Table 18 shows, GPRR is cheaper on our predictions than on the baselines', because its cost is driven by how much repair the prediction needs rather than by the size of the board. The realization cost is dominated by raster operations on the upsampled canvas and by the bridging search, both of which are restricted to the bounding box of the copper drawn for the net, so GPRR does not add a search over the whole board unless the fallback rerouting is triggered.

Table 16: **Route realization on the full test fold (1,968 nets).** Every realizer and encoder arm under the realization protocol we report: each learned decoder realizes the nets it can express, and the remaining 57, whose pad count falls outside the decoder's 2–8 pad budget, are realized by GPRR. Every column, segments per net included, therefore covers all 1,968 nets for every row, since the fallback is part of the pipeline being measured. These are the numbers quoted in the main text. Both context regimes are shown: reference-prefix gives each net the reference copper of the nets ordered before it, while AR rollout substitutes the model's own accumulated predictions. All rows use the same decoding budget and attach GPRR's repair stages, so connectivity is 1.000 throughout. Subscripts are the standard deviation over three diffusion seeds.

Realizer / encoder	Realized F1	Rout. (%)	DRVs	Seg./net	WL/GT	Vias/net
Reference-prefix context						
GPRR (hand-crafted)	$0.6002_{\pm 0.0024}$	$69.2_{\pm 0.5}$	$65.1_{\pm 1.9}$	$99.1_{\pm 1.6}$	$1.020_{\pm 0.002}$	$0.786_{\pm 0.011}$
WAND, by encoder						
DiT (ours)	$0.6019_{\pm 0.0016}$	$68.1_{\pm 1.0}$	$68.7_{\pm 1.5}$	$45.9_{\pm 2.0}$	$1.020_{\pm 0.001}$	$0.757_{\pm 0.015}$
DiT random-init	$0.6036_{\pm 0.0022}$	$66.2_{\pm 0.5}$	$70.4_{\pm 2.0}$	$46.5_{\pm 2.7}$	$1.011_{\pm 0.001}$	$0.719_{\pm 0.014}$
ViT encoder	$0.5674_{\pm 0.0023}$	$56.5_{\pm 0.4}$	$94.7_{\pm 2.7}$	$63.2_{\pm 1.2}$	$1.010_{\pm 0.002}$	$0.726_{\pm 0.009}$
DiT, no raster	$0.4721_{\pm 0.0007}$	$43.9_{\pm 0.3}$	$141.9_{\pm 1.6}$	$59.8_{\pm 1.1}$	$1.123_{\pm 0.002}$	$0.587_{\pm 0.015}$
ViT, no raster	$0.4540_{\pm 0.0009}$	$37.6_{\pm 0.1}$	$160.6_{\pm 1.4}$	$67.5_{\pm 1.0}$	$1.060_{\pm 0.001}$	$0.553_{\pm 0.013}$
AR rollout context						
GPRR (hand-crafted)	$0.5462_{\pm 0.0004}$	$75.2_{\pm 0.1}$	$55.6_{\pm 1.0}$	$103.4_{\pm 4.3}$	$1.036_{\pm 0.003}$	$0.885_{\pm 0.033}$
WAND, by encoder						
DiT (ours)	$0.5513_{\pm 0.0007}$	$72.8_{\pm 0.6}$	$59.5_{\pm 0.1}$	$47.3_{\pm 3.0}$	$1.032_{\pm 0.004}$	$0.845_{\pm 0.023}$
DiT random-init	$0.5531_{\pm 0.0019}$	$69.6_{\pm 0.2}$	$64.7_{\pm 0.3}$	$50.3_{\pm 3.1}$	$1.023_{\pm 0.002}$	$0.780_{\pm 0.008}$
ViT encoder	$0.5254_{\pm 0.0035}$	$57.4_{\pm 0.7}$	$93.7_{\pm 1.2}$	$66.1_{\pm 2.1}$	$1.015_{\pm 0.002}$	$0.771_{\pm 0.010}$
DiT, no raster	$0.4403_{\pm 0.0022}$	$43.0_{\pm 0.3}$	$154.4_{\pm 0.3}$	$61.2_{\pm 2.7}$	$1.386_{\pm 0.033}$	$0.700_{\pm 0.009}$
ViT, no raster	$0.4023_{\pm 0.0016}$	$34.2_{\pm 0.6}$	$171.7_{\pm 2.2}$	$67.1_{\pm 2.0}$	$1.341_{\pm 0.074}$	$0.589_{\pm 0.012}$

Table 17: **Route realization on the decoder-only population (1,911 nets).** The same arms as Table 16, restricted to the nets each decoder realizes itself, so no row contains any fallback geometry. This isolates the decoder from its fallback: routability is within 0.9 points of GPRR, and the segment count is 3.6× lower rather than 2.2×. The two tables agree on the ordering of every arm. Subscripts are the standard deviation over three diffusion seeds.

Realizer / encoder	Realized F1	Rout. (%)	DRVs	Seg./net	WL/GT	Vias/net
Reference-prefix context						
GPRR (hand-crafted)	$0.6051_{\pm 0.0025}$	$67.5_{\pm 0.3}$	$56.6_{\pm 1.4}$	$75.8_{\pm 1.3}$	$1.021_{\pm 0.003}$	$0.643_{\pm 0.008}$
WAND, by encoder						
DiT (ours)	$0.6068_{\pm 0.0017}$	$66.6_{\pm 1.1}$	$61.3_{\pm 1.4}$	$21.0_{\pm 0.5}$	$1.021_{\pm 0.001}$	$0.613_{\pm 0.007}$
DiT random-init	$0.6086_{\pm 0.0023}$	$64.8_{\pm 0.5}$	$62.2_{\pm 2.4}$	$21.6_{\pm 1.2}$	$1.012_{\pm 0.002}$	$0.574_{\pm 0.003}$
ViT encoder	$0.5712_{\pm 0.0023}$	$55.2_{\pm 0.4}$	$87.4_{\pm 2.8}$	$38.7_{\pm 0.5}$	$1.011_{\pm 0.002}$	$0.581_{\pm 0.009}$
DiT, no raster	$0.4732_{\pm 0.0006}$	$42.8_{\pm 0.3}$	$136.0_{\pm 1.1}$	$35.3_{\pm 0.6}$	$1.127_{\pm 0.002}$	$0.438_{\pm 0.005}$
ViT, no raster	$0.4545_{\pm 0.0008}$	$36.3_{\pm 0.1}$	$153.2_{\pm 0.9}$	$43.2_{\pm 0.7}$	$1.062_{\pm 0.001}$	$0.403_{\pm 0.005}$
AR rollout context						
GPRR (hand-crafted)	$0.5502_{\pm 0.0007}$	$72.1_{\pm 0.0}$	$50.3_{\pm 0.8}$	$79.3_{\pm 2.0}$	$1.038_{\pm 0.003}$	$0.726_{\pm 0.023}$
WAND, by encoder						
DiT (ours)	$0.5554_{\pm 0.0010}$	$70.3_{\pm 0.5}$	$54.4_{\pm 1.1}$	$21.5_{\pm 1.1}$	$1.034_{\pm 0.004}$	$0.685_{\pm 0.014}$
DiT random-init	$0.5573_{\pm 0.0022}$	$67.1_{\pm 0.3}$	$58.7_{\pm 0.7}$	$24.6_{\pm 0.8}$	$1.025_{\pm 0.002}$	$0.617_{\pm 0.015}$
ViT encoder	$0.5288_{\pm 0.0035}$	$56.0_{\pm 0.8}$	$88.0_{\pm 1.1}$	$40.9_{\pm 0.7}$	$1.017_{\pm 0.001}$	$0.609_{\pm 0.020}$
DiT, no raster	$0.4412_{\pm 0.0024}$	$41.8_{\pm 0.2}$	$149.4_{\pm 0.8}$	$35.8_{\pm 0.9}$	$1.399_{\pm 0.034}$	$0.535_{\pm 0.004}$
ViT, no raster	$0.4021_{\pm 0.0020}$	$32.9_{\pm 0.7}$	$167.1_{\pm 3.0}$	$41.9_{\pm 0.8}$	$1.352_{\pm 0.076}$	$0.422_{\pm 0.014}$

E Sampling Configuration Selection

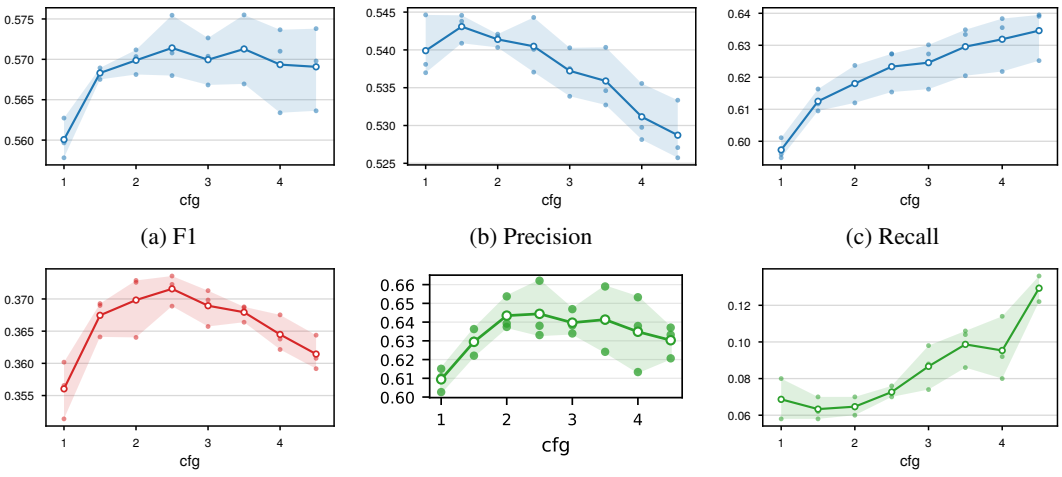
We select the sampling configuration on the validation split and apply it unchanged to the test set. We fix the denoising budget at 14 steps and sweep the classifier-free guidance (Ho & Salimans, 2022) scale from 1.0 to 4.5 over three random seeds.

Table 18: **Per-net generation and realization cost on the public benchmark.** Generation is model inference, timed for all arms in one sequential run on a single GPU. Realization is the mean runtime of GPRR and of full A⋆ rerouting on the same predictions, on the same CPU. A dash marks a stage that does not apply, since the reference and the uniform guide involve no model and the guidance-map baselines are realized by the A⋆ procedure only.

Model	Generation (ms / net)	GPRR (s / net)	A⋆ (s / net)
Reference	–	0.20	–
A⋆ (uniform)	–	–	0.17
VAE	0.21	–	2.17
UNet	2.34	–	0.25
cGAN-P	0.38	3.88	0.53
cGAN-H	0.29	3.80	0.55
VAE-H	0.20	6.46	2.97
DPM	50.09	5.46	4.24
DSB	32.14	5.70	1.04
RouteDiff	311.32	0.66	0.25

As shown in Figure 8, the tolerance-based F1 provides limited separation across guidance scales because increasing guidance decreases precision while increasing recall. Exact-match F1 and routability peak near 2.5, while the fallback rate is lowest around 1.5–2.0 and rises only modestly at 2.5. Based on these metrics, we use a guidance scale of 2.5 for all main experiments.

Compared with the earlier 28-step, CFG-4.5 configuration used during development, the selected setting also reduces generation time by 35% while improving exact-match F1, routability, and fallback frequency.



(a) F1 (b) Precision (c) Recall

(d) F1, exact (e) Routability (f) Fallback rate

Figure 8: **Validation performance across classifier-free guidance scales.** The tolerance-based F1 remains relatively stable because precision decreases as recall increases. Exact-match F1, routability, and GPRR fallback rate provide clearer separation, and exact-match F1 and routability motivate the selected scale of 2.5.

F Conditioning Analysis

F.1 Conditioning Pathway Ablation

To measure the contribution of each conditioning pathway, we retrain ***RouteDiff*** after removing the corresponding input rather than masking it only at inference. All variants use the same split, training

budget, sampling configuration, and checkpoint-selection protocol as the full model. All ablations use the prediction threshold selected for the full model.

Table 19 shows that removing any individual pathway reduces reference agreement. Target-pad tokens have the largest effect on F1, while removing the spatial fields causes the largest decrease in routability. The design-rule scalars behave differently from the other two pathways. Removing the design-rule scalars lowers reference agreement while leaving violations unchanged and slightly increasing routability (69.2% to 70.7%). This pattern is consistent with the controllability results in Sections F.2 and F.3: the scalar pathway primarily determines the drawn track width, whereas the clearance-aware field, which still encodes both rules and is retained in this ablation, carries most of the local feasibility information. Because GPRR reduces each prediction to a centerline before realization, differences in drawn width can affect raw F1 but not the realized geometry on which routability is measured. We do not attribute the small routability increase to a specific mechanism. Overall, the pathways provide complementary information rather than admitting a single importance ordering.

Table 19: **Conditioning ablations on the public benchmark.** Δ is F1 relative to ***RouteDiff***.

Conditioning	F1	Δ	Rout. (%)	DRVs
RouteDiff	**0.5965** $\pm$0.0020	–	**69.2** $\pm$0.5	**65.1** $\pm$1.9
− spatial fields	0.5671 $\pm$0.0020	-0.0294	65.6 $\pm$0.7	76.1 $\pm$0.4
− target-pad tokens	0.5609 $\pm$0.0027	-0.0356	68.7 $\pm$0.2	65.3 $\pm$1.1
− design-rule scalars	0.5804 $\pm$0.0040	-0.0161	70.7 $\pm$0.3	64.9 $\pm$0.5
w/o hetero. cond.	0.5356 $\pm$0.0060	-0.0609	62.8 $\pm$0.7	78.9 $\pm$1.8

F.2 Track-Width Controllability

We scale the requested track width by $\mu \in \{0.75, 1.5, 2, 3\}$ together with the uninstructed $\mu = 1$ while holding the random seed and remaining conditioning fixed. We measure the achieved ratio ρ relative to the uninstructed generation, so proportional compliance corresponds to $\rho = \mu$. The generated width of one net is estimated from its binarized per-layer trace mask as twice the mean of the upper quartile of the Euclidean distance transform, which approximates the width along the centreline, and the two copper layers are combined in proportion to their predicted pixel counts. The population is the 410 target nets whose requested width is at least three pixels on their own canvas, since a narrower request cannot be widened or narrowed distinguishably at 512×512. Because track width is provided through both a scalar rule and the clearance-aware spatial field, we intervene on each pathway separately and jointly.

The response is carried primarily by the scalar pathway. At $\mu = 3$, the scalar-only intervention reaches $\rho = 2.13$, compared with 2.17 when both pathways are modified and 1.16 when only the spatial field is changed. The response becomes sublinear at larger instruction scales, where the requested geometry increasingly departs from the routing configurations observed during training. Figure 9 shows qualitative examples under the joint intervention.

F.3 Clearance Controllability

Clearance specifies a lower bound rather than a target value, so routes that already satisfy an increased clearance need not change. We therefore evaluate only binding cases in which the uninstructed generation violates the requested clearance and the constraint is geometrically attainable. A case is binding when the tenth percentile of the local clearance of the uninstructed generation falls below the requested value. It is attainable when a trace of the requested width can still connect all target pads inside the region that clears the requested distance from other copper, which we test by bisecting on that distance and checking connectivity of the resulting region, treating layer changes optimistically. Cases that fail this test are reported separately as infeasible rather than as failures to comply. At the doubled instruction of Figure 5, 94 of the 615 swept nets are binding, of which 19 are infeasible and 75 form the recovery denominator. Recovery is measured as the fraction of these binding feasible cases that satisfy the increased clearance after intervention.

In contrast to track width, clearance responds primarily to the clearance-aware spatial field. The spatial intervention recovers substantially more binding violations than the scalar intervention, indicating that the spatial pathway provides the local feasibility information required to satisfy increased clearance. Figure 10 shows representative successful recoveries under the joint intervention, while Figure 5 reports the full quantitative population.

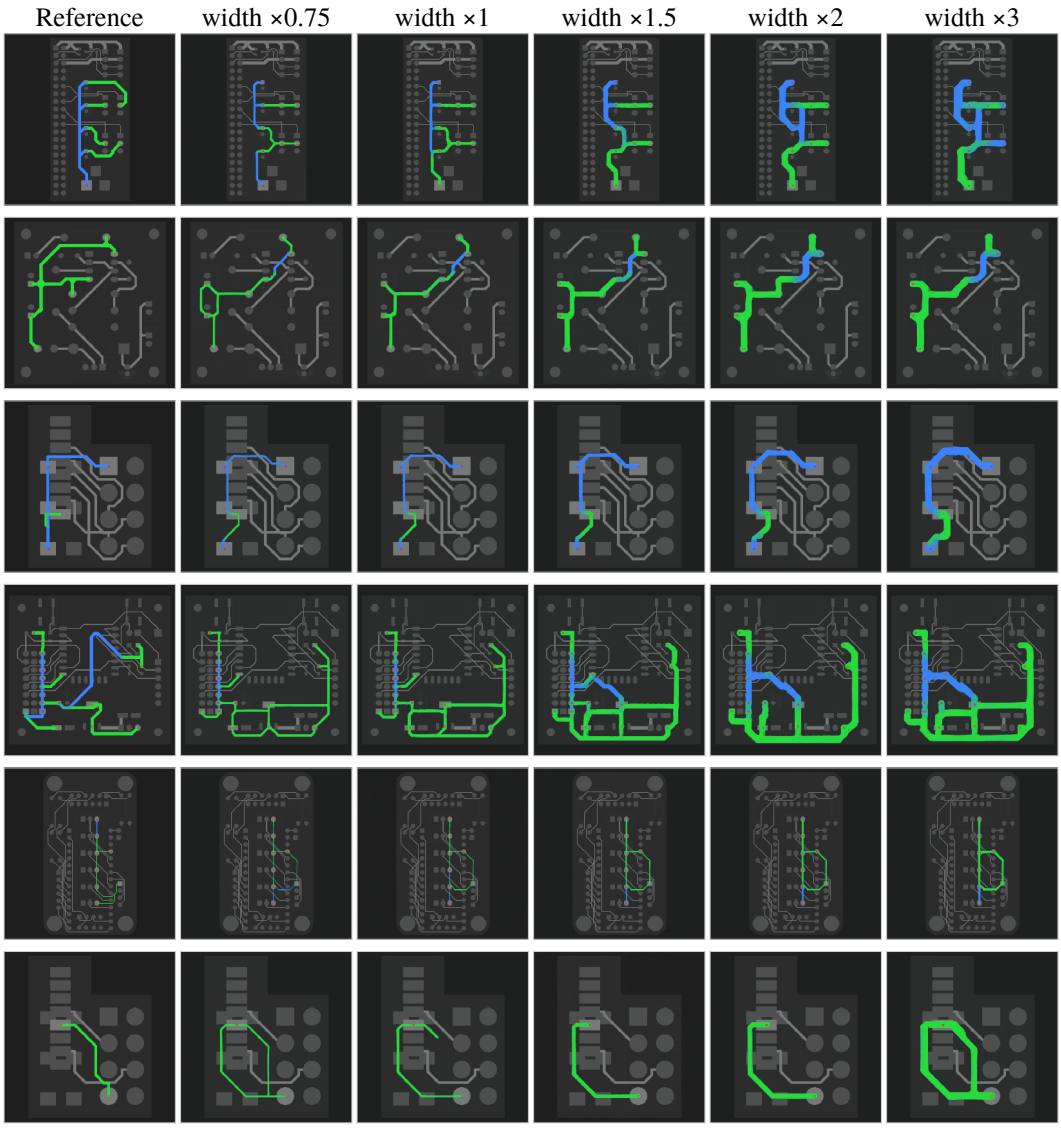


Figure 9: **Six held-out nets under a scaled width instruction.** Columns are the reference routing and generations at $\mu \in \{0.75, 1, 1.5, 2, 3\}$, with both the scalar and the clearance-aware field intervened together. Traces are the raw model output composited without a threshold. Green is the front layer, blue the back, and red circles mark the pads of the target net. Nets with more pins show the effect most clearly: a narrow instruction is met with thin detours, a wide one with a single thick trunk.

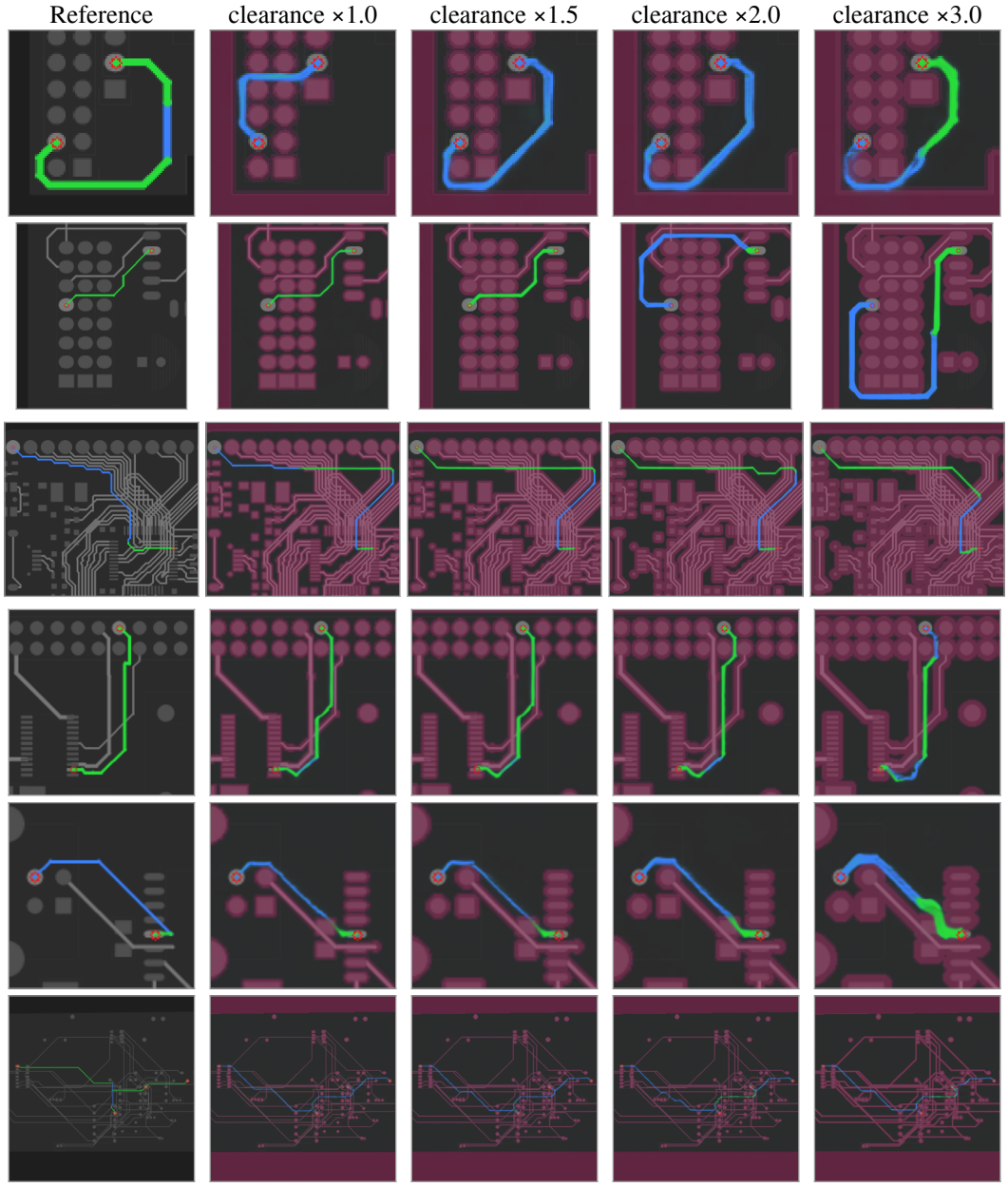


Figure 10: **Six held-out nets under a scaled clearance instruction.** Columns are the reference routing and generations at c scaled by $\{1.0, 1.5, 2.0, 3.0\}$, with both the scalar and the clearance-aware field intervened together. The $\times 1.0$ column is the uninstructed generation at the specified clearance. Rendering follows Figure 9: traces are the raw model output composited without a threshold, green is the front layer, blue the back, and red circles mark the pads of the target net. The magenta wash marks the instructed keep-out, the region within c of other copper, and the fainter wash the band that clears c but cannot hold the trace width. The remaining dark area is where a centreline may be placed. Both washes are drawn from the tighter of the two layers, so a route that avoids the constraint by changing layer can appear to enter them. All six nets are cases the instruction recovers. The population also contains nets it leaves unchanged and nets it worsens.