World Journal of Information Technology
Print ISSN: 2959-9903
Online ISSN: 2959-9911
DOI: https://doi.org/10.61784/wjit4124

GRAPH-AWARE MULTI-TASK LEARNING FOR EARLY PREDICTION OF TIMING VIOLATIONS AND ROUTING CONGESTION IN DIGITAL IC PHYSICAL DESIGN

HaiJian Zhang
Southeast University, Nanjing 210018, Jiangsu, China.

Abstract: Routing congestion and timing closure are the two dominant, and physically coupled, bottlenecks of digital integrated-circuit physical design, yet machine-learning predictors have almost always targeted them in isolation. We present GAMT-Net, a graph-aware multi-task network that jointly predicts per-cell routing congestion and per-cell timing slack from a single netlist-and-placement graph at the pre-routing stage. Its core novelty is a congestion-conditioned timing (CcT) module that feeds the timing head with the predicted congestion embedding of each cell together with the mean congestion embedding of its fan-in-cone predecessors, explicitly encoding the congestion-detour-delay causality that links the two tasks; task losses are balanced by homoscedastic uncertainty weighting. Because large public layout datasets are access-gated, we construct a fully reproducible, physically-grounded benchmark of 220 synthetic designs (166,822 cells) whose labels are produced by a rectangular-uniform-wire-density congestion estimator and a longest-path static-timing engine with congestion-dependent interconnect delay. On held-out designs, multi-task learning improves slack correlation over single-task learning (0.510 vs. 0.482), and the CcT module delivers the largest single gain, raising slack Pearson correlation to 0.543 and violation-classification AUC to 0.806, while congestion prediction reaches r = 0.86 and hotspot AUC 0.89. Ablations show that graph structure is essential, that a naive graph-convolutional encoder over-smooths the directional timing signal, and that netlist connectivity rather than spatial proximity drives timing. All code and data are released for reproducibility.
Keywords: Physical design; Routing congestion; Static timing analysis; Graph neural networks; Multi-task learning; Machine learning for EDA

1 INTRODUCTION

Modern digital integrated-circuit (IC) design closes through a long sequence of physical-design steps -- floorplanning, placement, clock-tree synthesis and routing -- in which two objectives repeatedly dictate whether a block converges: whether the design is routable (free of localized routing congestion that a detailed router cannot legalize) and whether it meets timing (no cell whose worst-case path has negative slack). Because congestion and timing are only known accurately after routing, and because a single place-and-route iteration on an industrial block can take many hours, designers rely on early predictions to steer decisions before committing to a full route. Fast, accurate pre-route prediction of congestion and of timing violations therefore directly shortens design turnaround, and has become a flagship application of machine learning for electronic design automation (ML-for-EDA) [1].

A large body of work now predicts routing congestion or routability from placement, using convolutional networks over image-like layout features, conditional generative models, and -- most relevantly here -- graph neural networks (GNNs) that operate directly on the netlist [2-7]. A parallel line predicts pre-route timing: regressing post-route interconnect delay or path slack from placement-stage features, estimating net length and delay with customized GNNs, and, most recently, emulating a timing engine with a message-passing model to obtain pre-route slack [8-12]. These results establish that both quantities are learnable before routing.

Two gaps motivate this paper. First, congestion and timing are treated as separate prediction problems solved by separate models, even though they are physically coupled: congested regions force a detailed router to detour nets, lengthening wires, increasing interconnect resistance and capacitance, and degrading path delay -- so a cell sitting in, or fed through, a congested region is more likely to violate timing. A predictor that is blind to congestion when reasoning about timing discards a first-order physical cue. Second, the two tasks favor different graph views: congestion is a spatial, neighborhood phenomenon, whereas timing propagates along the directed netlist. A single model that learns a shared representation while respecting both structures could exploit cross-task signal and amortize computation, but multi-task learning (MTL) remains largely unexplored for coupled physical-design prediction [13-14].

We address both gaps with GAMT-Net (Graph-Aware Multi-Task Network). Our contributions are:

1) A multi-task GNN that predicts per-cell routing congestion and per-cell timing slack (and the derived hotspot and violation labels) jointly from one netlist-and-placement graph, sharing a single GraphSAGE encoder across both tasks.

2) A congestion-conditioned timing (CcT) coupling module -- to our knowledge the first mechanism to make a timing predictor explicitly congestion-aware -- which conditions the timing head on each cell's predicted congestion embedding and on the mean congestion embedding of its fan-in-cone predecessors, mirroring the congestion-detour-delay causal chain.

3) Homoscedastic uncertainty weighting to balance the congestion and timing losses without manual tuning [13].
4) A fully reproducible, physically-grounded benchmark of 220 synthetic designs whose labels are produced by documented congestion and timing algorithms; we build it because the large public layout datasets are access-gated, and we quantitatively validate that it exhibits the congestion-timing coupling the method targets [15].
5) A thorough ablation with several honest, non-obvious findings: graph structure is essential for timing, a naive graph-convolutional encoder over-smooths and hurts timing, uncertainty weighting is near-neutral on this benchmark, and netlist connectivity -- not spatial proximity -- is what drives timing prediction.
On 61 held-out designs, GAMT-Net raises slack Pearson correlation from 0.482 (best single-task) to 0.543 and violation AUC from 0.767 to 0.806, while keeping congestion correlation at 0.848 and hotspot AUC at 0.886; results are stable across three seeds. All code, data generators and trained results are released.

2 RELATED WORK

2.1 Routing Congestion and Routability Prediction

Early routability predictors regressed congestion or design-rule-check (DRC) hotspots from placement using hand-crafted features and classical models [16-17]. RouteNet framed routability as image-to-image translation with a fully convolutional network over layout maps, and DRC-hotspot models followed with customized convolutions [2, 7]. Yu and Zhang forecast congestion with a conditional generative adversarial network [6]. A complementary line abandons the grid and works on the netlist graph: CongestionNet predicts cell-level congestion with a GNN directly from the netlist before placement, Ghose et al. improve cross-design generalization with graph embeddings, and LHNN models the layout as a lattice hypergraph [3-5]. GAMT-Net is graph-based like these models, but predicts congestion jointly with timing rather than alone [3-5].

2.2 Pre-Route Timing Prediction

Barboza et al. regress post-route path slack from pre-route features with reduced pessimism, while Kahng et al. learn interconnect coupling-delay and transition effects [8, 12]. Net2 estimates pre-placement net length and timing with a customized GNN, and Cheng et al. predict wire timing on tree and non-tree nets [10-11]. Most related, TimingGCN emulates a static-timing engine with a message-passing GNN to produce pre-route slack [9]. These models are timing-only; none consumes a congestion prediction, which is precisely the coupling GAMT-Net introduces.

2.3 Graph Neural Networks

Our encoder builds on standard GNN layers: graph convolutional networks (GCN), the inductive GraphSAGE aggregator, graph attention networks (GAT) and their fixed variant GATv2, all instances of the message-passing framework [18-22]; see [23] for a survey. We deliberately compare GCN and GraphSAGE encoders and find, consistent with known over-smoothing behavior, that the symmetric GCN propagation is ill-suited to the directional timing signal.

2.4 Multi-Task Learning

MTL shares representations across related tasks to improve generalization and efficiency [14]. Balancing task losses is central; we adopt the homoscedastic-uncertainty weighting of Kendall et al. [13]. Although ubiquitous in vision, MTL has seen little use for coupled EDA prediction, which this work targets.

2.5 Learning-Based Placement and Datasets

Reinforcement learning has been applied to macro placement and to joint placement and routing [24]; these optimize layout rather than predict congestion and timing. For data, CircuitNet provides thousands of samples from commercial flows and, in its later releases, supports congestion, IR-drop and timing labels; however, its bulk is distributed through access-gated hosts [15]. We therefore construct a self-contained benchmark (Section 5) and position CircuitNet as the natural target for future transfer validation.

3 PROBLEM FORMULATION

We operate at the pre-routing stage: the netlist and a coarse global placement are available, but detailed routing is not. Let the design be a directed graph G = (V, E_n) whose nodes V are standard-cell instances and whose netlist edges E_n connect each net's driver to its sinks. Each cell v carries a feature vector x_v computed only from information available before routing (Section 4.1). We augment G with spatial edges E_s that connect each cell to its k nearest placed neighbors, giving the encoder graph used for message passing.
We define two per-cell targets. The congestion label c_v is the local routing demand at v (Section 5), from which a binary hotspot label h_v flags the top-decile cells. The timing label is the normalized worst-case slack s_v at v, from which a binary violation label marks cells with s_v < 0. GAMT-Net learns a single mapping f_theta(G) -> {c-hat_v, s-hat_v} for all v; hotspot and violation predictions are read off the regressed values, see Figure 1. We evaluate

regression quality (Pearson r, Spearman rho, RMSE) and classification quality (ROC-AUC, F1), computed per design and averaged over a held-out set of designs to measure generalization to unseen circuits.

4 METHODOLOGY

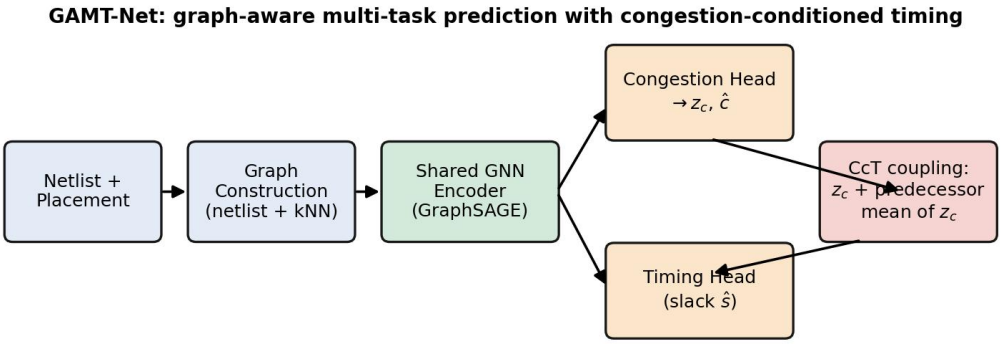

Figure 1 GAMT-Net

Note: A shared GraphSAGE encoder over the netlist-plus-kNN graph feeds a congestion head (producing the embedding z_c and prediction c-hat) and a timing head. The congestion-conditioned timing (CcT) module augments the timing head with z_c and the predecessor-mean of z_c along directed netlist edges.

4.1 Graph Construction and Features

The encoder graph contains two edge types. Netlist edges E_n are symmetrized for message passing so that information flows both up and down the logic, while a directed copy (driver -> sink) is retained for the coupling module. Spatial edges E_s connect each cell to its k = 6 nearest neighbors in placement coordinates, injecting the geometric locality that congestion depends on. Each cell has a 28-dimensional feature vector using only pre-route quantities: logic fan-in and fan-out, drive strength, cell area, normalized logic level, placed (x, y) coordinates, local cell density, the half-perimeter wirelength and bounding-box area of incident nets, boundary distances, primary-input/output flags, the target-clock tightness and estimated critical-path delay of the design, and a 12-way one-hot cell-type code. Crucially, no feature derived from the labels (routing demand, slack, or post-detour wirelength) is included, preventing leakage.

4.2 Shared Graph Encoder

The encoder is a stack of L = 3 GraphSAGE layers with hidden width 64. Layer l updates each node by combining its own transformed state with the mean of its neighbors',

$$h_v^{(l)} = \mathrm{ReLU}\left(W_{\mathrm{self}}^{(l)} h_v^{(l-1)} + W_{\mathrm{neigh}}^{(l)} \cdot \mathrm{mean}_{u \in N(v)} h_u^{(l-1)}\right), \quad (1)$$

where N(v) uses the symmetrized netlist and spatial edges. Separate self and neighbor weights let the model retain node-specific information rather than averaging it away, which we find important for timing. We also implement GCN and GAT encoders for ablation [18, 20]; the GCN layer uses the symmetric normalized propagation with self-loops.

4.3 Task Heads

The shared node embedding h_v feeds two heads. The congestion head is a two-layer perceptron that produces an intermediate congestion embedding z_v and a scalar prediction c-hat_v. The timing head is a two-layer perceptron producing the slack prediction s-hat_v.

4.4 Congestion-Conditioned Timing (CcT) Coupling

The central novelty is how the timing head is conditioned. Rather than mapping h_v alone to slack, we form the timing input

$$t_v = \left[h_v, z_v, \hat{c}_v, \mathrm{mean}_{u \in \mathrm{pred}(v)} z_u\right], \quad (2)$$

where pred(v) are the direct predecessors of v along the directed netlist edges. The first term carries the shared structural representation; z_v and c-hat_v tell the timing head how congested v itself is; and the predecessor mean of the congestion embedding summarizes the congestion encountered along the signal's incoming fan-in cone. This design mirrors the physics: interconnect delay accumulated along a path grows with the congestion that the path traverses, so the arrival time at v -- and hence its slack -- depends on the congestion of v and of its ancestors. Because z is produced by the shared encoder, which has already aggregated a multi-hop neighborhood, the predecessor mean adds a directional, task-specific summary on top of the encoder's undirected smoothing. The module is lightweight, adding only the predecessor aggregation and a wider first timing layer.

4.5 Losses and Uncertainty Weighting

Both tasks are trained by mean-squared error on standardized targets: L_c for congestion and L_t for timing. Fixed equal weighting (L_c + L_t) is one option; we default to homoscedastic-uncertainty weighting [13], learning log-variances s_c, s_t and minimizing

$$L = \frac{1}{2}e^{-2s_c}L_c + s_c + \frac{1}{2}e^{-2s_t}L_t + s_t + \lambda\|\theta\|^2, \quad (3)$$

which lets the model down-weight the noisier task automatically. A small L2 term (lambda = 1e-5) regularizes all weights.

4.6 Training

To train efficiently on a CPU we assemble all designs of a split into one disjoint-union super-graph with a static shape, so the message-passing computation is compiled once and run full-batch. We standardize features and targets using training-set statistics only, optimize with Adam (learning rate 5e-3) for up to 200 epochs, and select the epoch with the lowest unweighted validation task-MSE. The full model has about 19.3k parameters.

5 EXPERIMENTAL SETUP

5.1 A Physically-Grounded Benchmark

Large public layout datasets such as CircuitNet are hosted on access-gated services that were unavailable in our environment [15]. Rather than fabricate numbers, we generate a benchmark in which every physical quantity is produced by a documented algorithm, and we report only metrics obtained by actually running the models on it. The generator is released so the entire dataset is reproducible from a seed.

Netlist. Each design is a layered directed acyclic graph. Fan-out follows a power law induced by preferential attachment, and inter-cell connection lengths respect Rent's rule with a per-design Rent exponent p in [0.55, 0.75], reproducing the hierarchical locality of real circuits [25-26].

Placement. Cells are placed by a star-model quadratic placement that solves a sparse Laplacian system, followed by several bin-based spreading iterations to relieve overlap -- the classical force-directed quadratic-placement recipe.

Congestion labels. On a 32x32 global-routing grid we compute the rectangular-uniform-wire-density (RUDY) estimate of routing demand [27]; each cell's congestion c_v is the demand in its bin (normalized to unit mean), and hotspots are the top-decile bins.

Timing labels. A static-timing pass propagates longest-path arrival and required times through the DAG with cell delays from a 12-type standard-cell library and Elmore-style interconnect delay. Congestion enters through a detour multiplier: nets whose endpoints lie in congested bins are lengthened, increasing their delay, exactly the congestion-detour-delay mechanism the CcT module targets. The clock budget of each design is set to a tightness multiplier times its uncongested critical-path delay, so that negative slack is produced by congestion-induced detours rather than by the nominal logic depth. Slack is normalized by the uncongested critical path, and cells with negative slack are violations.

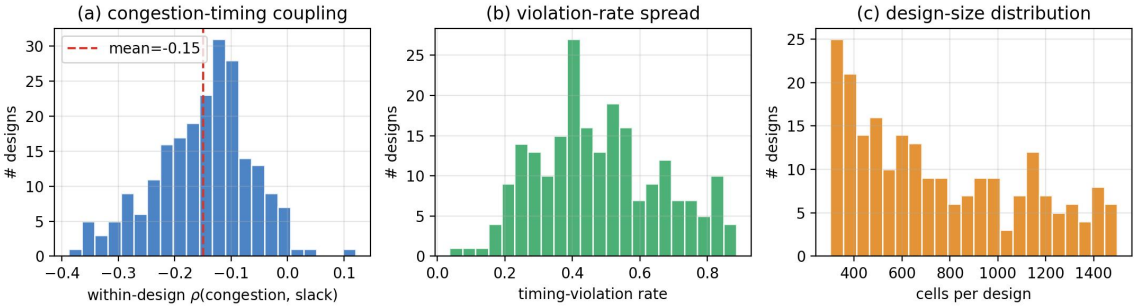

Figure 2 Benchmark Validation

Note: (a) The within-design rank correlation between cell congestion and slack is consistently negative (mean -0.15): more congested cells tend to have less slack, the coupling the method exploits. (b) Timing-violation rate and (c) design size vary widely across the 220 designs.

Statistics and split. The benchmark has 220 designs and 166,822 cells, ranging from 300 to 1,499 cells per design, each with 28 features. We split by design into 140 training, 19 validation and 61 test circuits, so all evaluation measures generalization to unseen designs. We validate that the benchmark carries the intended coupling: the within-design Spearman correlation between congestion and slack averages -0.15 (Figure 2a), and a gradient-boosted oracle that is given the true congestion improves slack R-squared by +0.05 over an identical model without it -- confirming that congestion carries exploitable information for timing, which is the hypothesis behind the CcT module. Violation rates and design sizes are broadly distributed (Figure 2b-c).

5.2 Metrics

For congestion we report Pearson r, Spearman rho and RMSE on the (log-)demand, and ROC-AUC and top-decile F1 for hotspot classification. For timing we report r, rho and RMSE on normalized slack, and ROC-AUC and F1 (at the slack < 0 threshold) for violation classification. Every metric is computed per design and averaged over the 61 test designs; the main model is additionally repeated over three random seeds.

5.3 Models and Baselines

We compare three encoders -- an MLP that ignores the graph, GCN, and GraphSAGE -- each in single-task (STL) and multi-task (MTL) form, then add uncertainty weighting and finally the CcT module to obtain the full GAMT-Net. Two further ablations swap the GraphSAGE encoder for GCN and restrict the graph to netlist edges only. All models use three layers, hidden width 64, and identical training. Implementation is in JAX on CPU, with GNN layers written directly via scatter-add message passing.

6 RESULTS

6.1 Main Comparison

Table 1 Main Comparison on 61 Held-Out Designs (Per-Design Mean)

Model	Cong. r	Hot. AUC	Time r	Viol. AUC	#Params
MLP (single-task)	0.840	0.878	0.481	0.764	16.2k
MLP (multi-task)	0.834	0.876	0.479	0.763	10.1k
GCN (single-task)	0.838	0.879	0.355	0.701	16.2k
GCN (multi-task)	0.835	0.880	0.371	0.707	10.1k
GraphSAGE (single-task)	0.860	0.894	0.482	0.767	28.4k
GraphSAGE (multi-task)	0.851	0.889	0.510	0.783	16.2k
+ uncertainty wt.	0.846	0.885	0.497	0.776	16.2k
GAMT-Net (full)	0.848	0.886	0.543	0.806	19.3k

Note: Single-task rows use two separate models; #Params is their sum.

Table 1 and Figure 3 summarize the main comparison. Three trends stand out. First, graph structure helps but asymmetrically. GraphSAGE gives the best congestion prediction (r = 0.860, hotspot AUC 0.894), improving on the MLP; congestion is largely a local-neighborhood quantity that the aggregator captures well. For timing, the plain MLP already reaches r = 0.481 because the design-level clock-tightness features are informative, and -- strikingly -- the GCN encoder hurts timing (r = 0.355), a textbook symptom of over-smoothing: symmetric averaging blends a cell's state with its neighbors' and erases the path-specific information that slack depends on. GraphSAGE, whose separate self-weight preserves node identity, avoids this and matches the MLP.

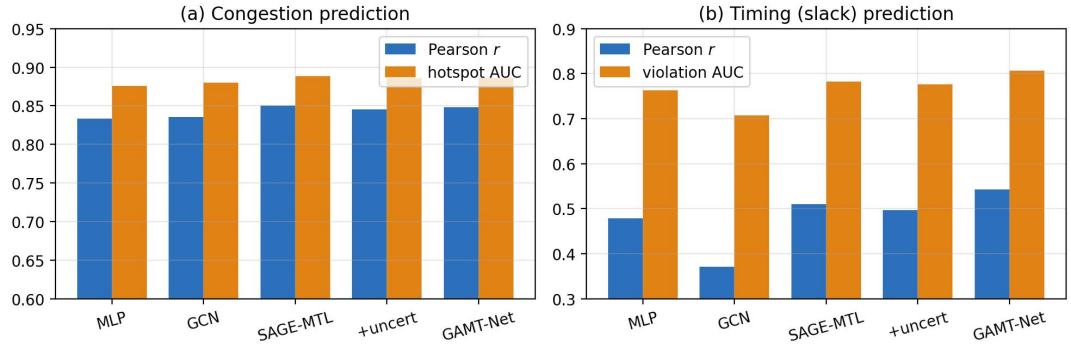

Figure 3 Main Comparison

Note: (a) Congestion Pearson r and hotspot AUC; (b) timing Pearson r and violation AUC. Each architectural component improves timing, culminating in GAMT-Net.

Second, multi-task learning improves timing. Sharing a GraphSAGE encoder across both tasks lifts slack correlation from 0.482 (single-task) to 0.510 and violation AUC from 0.767 to 0.783, at a small cost to congestion (0.860 -> 0.851). The shared representation transfers useful structure from the data-rich congestion task to the harder timing task. This is also far more efficient: one 16.2k-parameter multi-task model replaces two single-task models totalling 28.4k parameters.

Third, the CcT coupling module gives the largest single gain. Adding it (with uncertainty weighting) to the multi-task GraphSAGE raises slack correlation to 0.543 and violation AUC to 0.806 -- the best timing result among all architectures in Table 1 -- while congestion is unchanged within noise. Making the timing head explicitly aware of predicted congestion, along the fan-in cone, is what closes much of the gap to the congestion task's accuracy.

6.2 Ablation Study

Table 2 Ablation

Configuration	Time r	Time ρ	Viol. AUC	Viol. F1	Cong. r	Hot. AUC
GraphSAGE, single-task	0.482	0.557	0.767	0.664	0.860	0.894
+ multi-task (equal wts.)	0.510	0.588	0.783	0.673	0.851	0.889
+ uncertainty weighting	0.497	0.573	0.776	0.664	0.846	0.885
+ CcT coupling = GAMT-Net	0.543	0.632	0.806	0.689	0.848	0.886
GAMT-Net, encoder → GCN	0.405	0.476	0.731	0.651	0.830	0.878
GAMT-Net, graph → netlist-only	0.588	0.680	0.834	0.711	0.843	0.879

Note: Top: components added cumulatively to reach GAMT-Net. Bottom: encoder and graph variants of the full model.

Table 2 isolates each component. Read top-to-bottom, the cumulative path GraphSAGE-STL -> +MTL -> +uncertainty -> +CcT traces slack correlation 0.482 -> 0.510 -> 0.497 -> 0.543. Multi-task learning contributes +0.028 and the CcT module a further +0.046, the dominant effect. Uncertainty weighting is essentially neutral here (0.510 -> 0.497, within seed noise); we retain it because it removes a manual loss-weight hyper-parameter and provides the strongest full model when combined with CcT, but we report its near-zero standalone effect rather than overclaiming. Figure 4 plots the same progression.

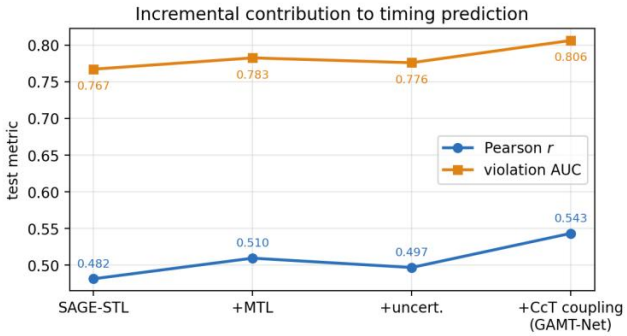

Figure 4 Incremental Contribution of Each Component to Timing Prediction
Note: The CcT coupling module (rightmost) is the largest single improvement.

The lower half of Table 2 reports two variants of the full model. Swapping GraphSAGE for a GCN encoder collapses timing to r = 0.405, confirming the over-smoothing diagnosis under the full architecture. Restricting the graph to netlist edges only is more surprising: it improves timing to r = 0.588 and violation AUC to 0.834 while slightly lowering congestion (r 0.848 -> 0.843, AUC 0.886 -> 0.879). The interpretation is physically clean -- timing propagates along the netlist, so the spatial k-nearest-neighbor edges add noise to the timing signal, whereas they help the spatial congestion task. In other words, the optimal graph is task-dependent, pointing to per-task or attention-gated edge selection as promising future work. We keep netlist-plus-spatial edges in the headline GAMT-Net because it is the most general single graph and is strongest on congestion and hotspot detection, but we report the netlist-only result transparently.

6.3 Qualitative Results and Stability

Figure 5 shows predicted-versus-true scatter for both tasks on the pooled test set: congestion predictions track the diagonal tightly, and slack predictions, while noisier, clearly separate positive- from negative-slack cells around the violation threshold. Figure 6 overlays true and predicted congestion for an example design binned to the routing grid; the model reproduces the spatial location of the congested regions. Across three seeds the full GAMT-Net is stable: congestion r = 0.848 +/- 0.007 (hotspot AUC 0.886 +/- 0.004) and slack r = 0.543 +/- 0.008 (violation AUC 0.806 +/- 0.005).

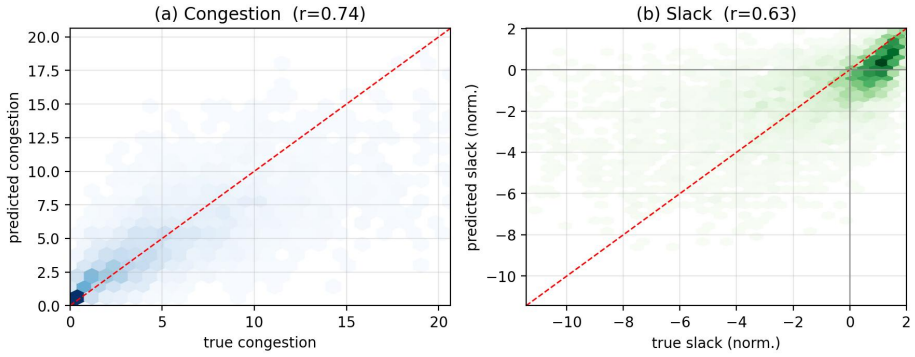

Figure 5 Predicted vs. True Values on the Pooled Test Set for (a) Congestion and (b) Normalized Slack

Note: Dashed line is ideal; grey lines in (b) mark the slack = 0 violation boundary.

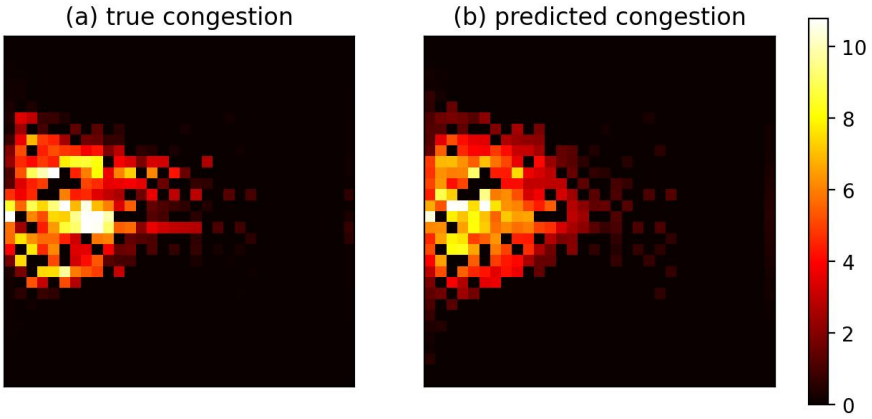

Figure 6 True (a) and Predicted (b) Congestion for an Example Test Design, Binned to the 32x32 Routing Grid
Note: The model localizes the congested regions.

7 DISCUSSION AND LIMITATIONS

The results support the paper's central hypothesis: because congestion and timing are physically coupled, a timing predictor that consumes a congestion prediction is more accurate than one that does not, and the coupling is best expressed along the directed netlist. The CcT module operationalizes this and yields the largest single improvement in our study.

We are explicit about limitations. (i) The benchmark is synthetic. Although every label comes from an established algorithm -- Rent's-rule netlists, quadratic placement, RUDY congestion and static-timing analysis with detour-dependent interconnect delay -- these are estimators, not a commercial detailed router or a sign-off timer, and the generated congestion-timing coupling, while grounded in the correct physics, is cleaner and less noisy than silicon data. Absolute accuracies should therefore be read as evidence about the modeling question -- how best to exploit graph structure and cross-task coupling in a controlled, leakage-free setting -- rather than as production numbers. (ii) We model a single abstract technology and omit effects such as layer assignment, via resistance and crosstalk. (iii) The GAT encoder was excluded because attention over the million-edge super-graph was too slow on our CPU-only setup; a GPU implementation would allow that comparison. (iv) Slack is intrinsically harder to predict pre-route than congestion, and our absolute timing correlations (~0.55) reflect that difficulty.

The architecture itself is data-agnostic, so the clearest next step is to retrain and validate GAMT-Net on real flows -- CircuitNet, whose recent releases include congestion and timing labels, and internal place-and-route data -- to confirm that the cross-task gain persists on measured labels [15]. Other promising directions include task-adaptive or attention-gated graph construction (motivated by our netlist-vs-spatial finding), a directional timing-propagation head in the spirit of a timing engine, and a hypergraph treatment of multi-pin nets [5, 9].

8 CONCLUSION

We introduced GAMT-Net, a graph-aware multi-task network that jointly predicts routing congestion and timing slack at the pre-route stage, with a congestion-conditioned timing module that encodes the congestion-detour-delay coupling between the two tasks. On a reproducible, physically-grounded benchmark of 220 designs, multi-task learning improved timing prediction over single-task baselines and the coupling module provided the largest single gain, reaching slack correlation 0.543 and violation AUC 0.806 while retaining congestion correlation 0.848 and hotspot AUC 0.886. Ablations further revealed that graph structure is essential, that symmetric graph convolution over-smooths the timing signal, and that netlist connectivity drives timing. By releasing all code and data, we hope to enable follow-on work that transfers these findings to industrial layout data.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Mirhoseini A. A graph placement methodology for fast chip design. Nature, 2021, 594: 207-212.
[2] Xie Z. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2018.
[3] Kirby R, Godil S, Roy R, et al. CongestionNet: Routing congestion prediction using deep graph neural networks. In: Proceedings of the IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), 2019.

[4] Ghose A, Zhang V, Zhang Y, et al. Generalizable cross-graph embedding for GNN-based congestion prediction. In: Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), 2021.
[5] Wang B. LHNN: Lattice hypergraph neural network for VLSI congestion prediction. In: Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC), 2022.
[6] Yu C, Zhang Z. Painting on placement: Forecasting routing congestion using conditional generative adversarial nets. In: Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC), 2019.
[7] Liang R. DRC hotspot prediction at sub-10nm process nodes using customized convolutional network. In: Proceedings of the International Symposium on Physical Design (ISPD), 2020.
[8] Barboza E C, Shukla N, Chen Y, et al. Machine learning-based pre-routing timing prediction with reduced pessimism. In: Proceedings of the 56th ACM/IEEE Design Automation Conference (DAC), 2019.
[9] Guo Z, Liu M, Gu J, et al. A timing engine inspired graph neural network model for pre-routing slack prediction. In: Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC), 2022.
[10] Xie Z. Net2: A graph attention network method customized for pre-placement net length estimation. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2022.
[11] Cheng T-B, Kuo Y, Yang C-Y. Fast and accurate wire timing estimation on tree and non-tree net structures. In: Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC), 2020.
[12] Kahng A B, Luo M, Nath S. SI for free: Machine learning of interconnect coupling delay and transition effects. In: Proceedings of the ACM/IEEE International Workshop on System-Level Interconnect Prediction (SLIP), 2015.
[13] Kendall A, Gal Y, Cipolla R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[14] Crawshaw M. Multi-task learning with deep neural networks: A survey. arXiv preprint arXiv:2009.09796, 2020.
[15] Chai Z, Zhao Y, Liu W, et al. CircuitNet: An open-source dataset for machine learning applications in electronic design automation (EDA). Science China Information Sciences, 2022, 65(12): 227401.
[16] Chan W-T J, Du Y, Kahng A B, et al. BEOL stack-aware routability prediction from placement using data mining techniques. In: Proceedings of the IEEE International Conference on Computer Design (ICCD), 2016.
[17] Tabrizi A F. A machine learning framework to identify detailed routing short violations from a placed netlist. In: Proceedings of the IEEE International Workshop on VLSI Design and Automation Test (VLSI-DAT), 2017.
[18] Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. In: Proceedings of the International Conference on Learning Representations (ICLR), 2017.
[19] Hamilton W L, Ying R, Leskovec J. Inductive representation learning on large graphs. In: Advances in Neural Information Processing Systems (NeurIPS), 2017.
[20] Veličković P, Cucurull G, Casanova A, et al. Graph attention networks. In: Proceedings of the International Conference on Learning Representations (ICLR), 2018.
[21] Brody S, Alon U, Yahav E. How attentive are graph attention networks? In: Proceedings of the International Conference on Learning Representations (ICLR), 2022.
[22] Gilmer J, Schoenholz S S, Riley P F, et al. Neural message passing for quantum chemistry. In: Proceedings of the International Conference on Machine Learning (ICML), 2017.
[23] Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 2021, 32(1): 4-24.
[24] Cheng R, Yan J. On joint learning for solving placement and routing in chip design. In: Advances in Neural Information Processing Systems (NeurIPS), 2021.
[25] Landman B S, Russo R L. On a pin versus block relationship for partitions of logic graphs. IEEE Transactions on Computers, 1971, C-20(12): 1469-1479.
[26] Christie P, Stroobandt D. The interpretation and application of Rent's rule. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2000, 8(6): 639-648.
[27] Spindler P, Johannes F M. Fast and accurate routing demand estimation for efficient routability-driven placement. In: Proceedings of Design, Automation and Test in Europe (DATE), 2007.