TopoGeoNet: A Scalable Topological and Geometric Learning Framework for Spatial Graphs

Zhishang Luo[1], **Zhili Xiong**[2], **Samantha Chen**[1], **David Z. Pan**[2], **Yusu Wang**[1]
[1]University of California, San Diego
[2]University of Texas at Austin
{zluo, samchen, yusuwang}@ucsd.edu, {zhilix, dpan}@utexas.edu

Abstract

Spatial graphs – graphs whose nodes are associated with geometric coordinates – arise in diverse domains including 3d point clouds, molecular structures, chip circuits and geospatial terrains. Different from generic graphs, learning on spatial graphs requires capturing both topological structure (graph connectivity) and the geometric context (spatial layout). We propose **TopoGeoNet**, a scalable hybrid architecture that integrates message passing in graph neural networks (GNNs) with a multi-scale Convolutional U-Net over spatial grids to let information flow seamlessly both locally (via graph neighborhoods) and over global spatial grids. This heterogeneous design allows the model to jointly learn topological and geometric information while also decoupling geometric aggregation and topological propagation. It allows the model to learn multi-scale geometric context and capture long-range interactions effectively and efficiently. We apply TopoGeoNet to two challenging large-scale spatial graph problems, chip circuits congestion prediction and terrain shortest path distance prediction, graphs ranging from $100K$ to $4M$ nodes. We showed that TopoGeoNet achieves state-of-the-art accuracy on both tasks compared to various SoTA architectures, demonstrating the power of unified geometric-topological learning in large-scale spatial graphs. Dataset and codes are available at https://github.com/luckyjackluo/TopoGeoNet.

1 Introduction

Large-scale spatial graphs—where nodes and edges are embedded in $\mathbb{R}^d$ (e.g., cell placements in chips; sampled vertices on terrains)—arise across diverse domains. Here *large-scale* refers to graphs that each can contain several million of nodes, rather than a large number of small graphs as in most of the molecular benchmarks. Predictive tasks on such graphs require efficient learning of both *topological structure* and *geometric context*, and often rely on capturing *long-range* interactions.

While Message Passing Neural Networks (MPNNs) [1, 2] scale nearly linearly with graph size in terms of number of nodes $|V|$ and edges $|E|$, they struggle to propagate information across long distances due to *over-smoothing* [3, 4] and *over-squashing* [5, 6]. Consequently, learning both global geometric context and local topological interactions in million-node spatial graphs remains challenging.

Prior approaches. (i) *Geometry-aware GNNs* (e.g., EGNN, SE(3)-GNN) enforce Euclidean equivariances and perform well on molecular and point-cloud tasks [7, 8], but still rely on local message passing or attention and do not scale to million-nodes settings. (ii) *Embedding/k-Nearest-Neighbor* methods such as Geom-GCN connect distant nodes via latent neighborhoods [9], while constructing and updating these neighborhoods becomes prohibitive at million-nodes scale. (iii) *Graph Transformers* (Graphormer; GraphGPS) use attention mechanisms [10–12] to overcome message-passing bottlenecks. However, the quadratic attention complexity still restricts them to graphs with millions of nodes. (iv) In chip design specifically, *CNN/UNet-style grid models* can scale to million-nodes and capture spatial patterns [13, 14], but they usually ignore the explicit netlist connectivity compared to

, TopoGeoNet: A Scalable Topological and Geometric Learning Framework for Spatial Graphs (Extended Abstract). Presented at the Fourth Learning on Graphs Conference (LoG 2025), Hybrid Event, December 10–12, 2025.

MPNN and we show that it can have inferior empirical performances compared to graph-learning-based models.

Scalable MPNNs with virtual nodes. Recent works introduce virtual nodes to improve scalability. *SMPNN* [15] replaces global attention with point-wise feedforward layer as virtual nodes to aggregate global information. However, it is challenging for single-scale virtual nodes (which aggregate global information) to aggregate in a multi-scale manner. *RouteGNN* [16] combines hypergraph neural networks with partition-level grid virtual nodes, but their single-scale design limits on local cross-partition communication and miss the more global information. *DE-HNN* [17] introduces hierarchical virtual nodesbut requires external partitioning and manual tuning of intermediate scales.

This work: TopoGeoNet. We introduce *TopoGeoNet*, a unified architecture that *decouples geometric* multi-scale aggregation from *topological* local propagation within each layer. Given any spatial (hyper)graph $G = (V, E)$ with node coordinates $\{p_v \in \mathbb{R}^d\}$ and *arbitrary* topology E (e.g., highly irregular netlists with long-range, non-metric edges), **TopoGeoNet** performs *Node→Grid→Node*: node embeddings are projected to a fixed-resolution *auxiliary* grid (size $(W \times H)$ is a model hyperparameter), processed by a U-Net to capture hierarchical long-range geometric context, then reassigned to nodes and propagated along the *original* (hyper)graph via an MPNN. Crucially, the grid is a *model-side buffer* defined only by $\{p_v\}$ and (W, H)—it does *not* assume or impose grid-structured inputs; message passing always operates on (V, E) unchanged. This integration yields global receptive fields (U-Net) and local fidelity (MPNN) without the memory/computation costs of full attention. On massive spatial graphs, **TopoGeoNet** is effective and efficient: for terrain SPD (up to 4M nodes) it reduces Mean Relative Error by 27% vs. SMPNN [15]; for chip congestion prediction (up to 1M nodes) it improves accuracy by 8.3% over RouteGNN [16] while being 24% faster at inference. Integrated into a routability-driven placement loop, **TopoGeoNet** improves a congestion score by 16% over a no-ML baseline and by 11% over the strongest prior ML-assisted placer [16]. Full architecture/training details, ablations (grid size/depth), runtime/memory, equivariance diagnostics, and data release appear in the Appendix.

2 Methodology: TopoGeoNet

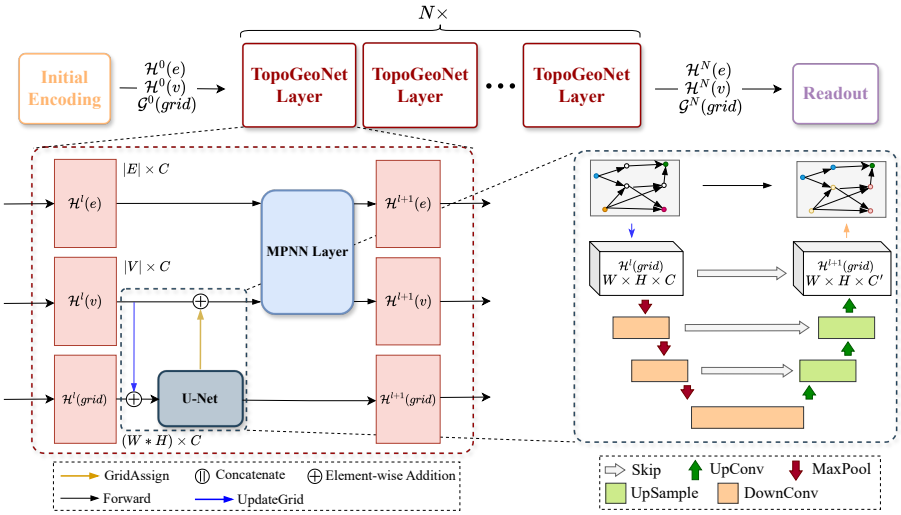

Figure 1: TopoGeoNet framework. **Figure 1(a)** on the left illustrates the overall architecture of **TopoGeoNet**, consisting of multiple TopoGeoNet layers. Each layer contains two stages: (i) Node embeddings are enriched by a U-Net module. (ii) The enriched embeddings are processed to a MPNN layer. **Figure 1(b)** on the right details the U-Net module: node embeddings are projected onto $grids$, processed by U-Net to capture multi-scale geometric context, and then reassigned back to nodes.

Spatial (hyper)Graph. We define a spatial graph (or hypergraph) $G = (V, E)$ where each node $v \in V$, besides other features, is associated with a d-dimensional coordinate vector $\mathbf{p}_v \in \mathbb{R}^d$

representing its spatial location. For simplicity, we assume that all nodes lie within a bounded region:

$$\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}], \quad (x_{\min}, y_{\min}) = \min_{v \in V} \mathbf{p}_v, \ (x_{\max}, y_{\max}) = \max_{v \in V} \mathbf{p}_v.$$

Grids. Before the initial encoding, to better capture the geometric context of the spatial graph, we partition bounded region Ω into $W \times H$ axis-aligned cells, and we represent each cell as a $grid \in \mathcal{G}$, see Figure 1. Before Initial Encoding, our **TopoGeoNet** will first create grid features by aggregating the node features from the nodes to grids via UPDATEGRID, see Algorithm 1, in Appendix A. Unlike clustering-based virtual-node hierarchies (e.g., METIS/K-means + pooling), our grid is a model-side buffer computed in $\mathcal{O}(|V|)$, and thus faster than most of the clustering based methods. We show empirical results that clustering methods have significantly longer run-time with no accuracy benefit, see Table 9 in Appendix C.2.

Initial Encoding. Each node coordinate $\mathbf{p}_v$ is encoded with a **Fourier Positional Encoder** and concatenated with other node features. We show the performance improvement with Fourier encoder in Appendix C.2.1. A MLP maps the concatenated features to initial node embeddings $\mathcal{H}^0(v)$. Separate MLPs generate initial edge embeddings $\mathcal{H}^0(e)$ and initial grid embeddings $\mathcal{H}^0(grid)$ from the features as described above, see Figure 1.

Node–Grid Communication. Before each MPNN layer, similar to what we did during *Initial Encoding*, node embeddings are aggregated to spatial grids $\mathcal{G}$ via UPDATEGRID. These grid embeddings are processed by a U-Net, which enriches them with multi-scale geometric context before being reassigned back to nodes via GRIDASSIGN, see details of both functions in Algorithm 1, in Appendix A. We show that, even without further integration with U-Net, the addition of grid-based local virtual nodes significantly improves performance, see Table 10, in Appendix C.2.

U-Net as Hierarchical Virtual Nodes. Figure 1(b) shows how U-Net is integrated. The encoder applies *DownConv* blocks with *MaxPool*, progressively coarsening the grid resolution ($H \times W \to \frac{H}{2} \times \frac{W}{2}$) while expanding receptive fields. At depth k, features cover regions of size $\sim 2^k$ per axis, culminating in a bottleneck that encodes strong global context [18]. The decoder mirrors this by successively doubling resolution via *UpConv* (transposed convolution), and at each scale concatenates the upsampled features. This top-down pathway with fused multi-scale information similar to feature pyramids and encoder–decoder refinements [19, 20].

Unlike static hierarchical virtual nodes (e.g., fixed partitions in DE-HNN [17]), U-Net *learns* which patterns and structures are most informative at each resolution. This adaptive aggregation allows the model to emphasize relevant global or local structures depending on the task, yielding more expressive multi-scale representations for global structures and long-range interactions [21–23]. A comparison between our grid projection and conventional graph-pooling schemes, as well as ablations on grid resolution and message-passing depth, can be found in Appendix C.2.

Message Passing. After node–grid communication, each node embedding $\mathcal{H}^l(v)$ has already been enriched with global multi-scale context from the U-Net via GRIDASSIGN. Depending on if the problem has spatial graph G or spatial hypergraph H as input, we then use different types of MPNN layer for further node embeddings update with graph topology. For the terrain SPD problem, we use *Graph Attention Network* [24] to update node embeddings $\mathcal{H}^l(v)$ via local graph connectivity between nodes at each layer l. For the chip congestion problem, we use *Directed Equivariant Hypergraph Neural Network* (DE-HNN) [17] to update the node embeddings $\mathcal{H}^l(v)$ and edge (net) embeddings $\mathcal{H}^l(e)$ via connectivity defined by each hyperedge e at each layer l, see Figure 1(a).

Complexity and Scalability. Each TopoGeoNet layer consists of three components with distinct computational costs:

$$\text{MPNN: } \mathcal{O}(|E|d), \quad \text{UPDATEGRID} + \text{GRIDASSIGN: } \mathcal{O}(|V|d), \quad \text{U-Net: } \mathcal{O}(WHd),$$

where d is the embedding dimension, and $W \times H$ is the grid resolution. Hence, the total per-layer cost is

$$\mathcal{O}(|E|d + |V|d + WHd),$$

which scales *linearly* in graph size ($|V|+|E|$) and adds a grid term independent of graph topology. This ensures near-linear scalability across diverse spatial graph structures. See the empirical run-time comparisons with other baseline models in Table 2 and Table 1.

Readout. For terrain SPD, we use a Siamese network on the final node embeddings $\mathcal{H}^L(v)$ for distance estimation. For chip congestion prediction, we add one more U-Net after the final TopoGeoNet layer, and pass the output grid embeddings $\mathcal{H}^{L+1}(grid)$ to a MLP to get final prediction align with the shape of congestion map.

3 Experiments

3.1 Experiment Setup

Chip Congestion Problem. Following prior works [16, 17, 25], we represent a chip circuit as a hypergraph $H = (V, E)$ where V is the set of cells and E is the set of hyperedges (nets). Each node $v \in V$ has a 2D placement coordinate (x, y) and size $\mathbf{s}_v = (w_v, h_v)$; each hyperedge $e \in E$ is a subset of cells. The task is to predict the scalar congestion value for each spatial cell $grid$, indicating routing overflow within that region. We use 12 large designs from the ISPD'16 contest [26], ranging from 100K to 1M nodes. Ground-truth congestion maps are obtained with modern chip design tools [27, 28]. We compare against several strong baselines, containing SoTA in terms of both scalable GNN and for netlists: (i) the image-based *pix2pixHD* model [29] that uses only grid features; (ii) *RouteGNN* [16], the previous SoTA that combines hypergraphs with grid features; and (iii) *SMPNN* [15], a scalable model that introduces a pointwise feedforward layer; (iv) *full DE-HNN* [17], a hypergraph neural network with hierarchical virtual nodes structure.

Terrain SPD Problem. We also evaluate on shortest-path distance (SPD) prediction over terrains. Each terrain graph is constructed as a grid graph from LiDAR elevation models. Each node corresponds to a terrain point with (x, y, z) coordinate, and is connected to its 8 neighbors; edges are weighted by Euclidean distance. We test on three terrains: Norway (4M nodes), Holland (1M), and Philadelphia (1M). We compare against: (i) *GAT* [24]; (ii) *Coarse-GAT* [30], which trains on downsampled graphs and then evaluates on the full-scale terrain; and (iii) *SMPNN* [15]. We attempted Graph Transformers (*Graphormer* [11], *GraphGPS* [31]), but both failed with out-of-memory errors on our million-scale graphs (see device details in Appendix C.1).

We also tested the robustness of our **TopoGeoNet** on perturbed irregular Norway terrain graphs and small-graph QM9 appears in App. C.3; TopoGeoNet holds gains on terrain graphs with grid structures destructed and remains competitive on small graphs.

3.2 Model Training and Prediction

Chip Congestion Problem. We report average performance using three metrics: *Normalized Root-Mean-Square-Error* (NRMS, lower better), *Structural-Similarity-Index-Measure* (SSIM, higher better), R^2 (higher better) and inference run-time per placement (in seconds, lower better), see Table 1. RouteGNN substantially outperforms pix2pixHD, suggesting the importance of combining the learning of circuit topology and geometric context. Our **TopoGeoNet** achieves the best overall performance. Beyond prediction accuracy, we integrate **TopoGeoNet** with chip design tools [27, 28] to evaluate its utility in the congestion (routability) driven placement optimization. We observe consistent improvements: **TopoGeoNet** reduces congestion scores by 16% on average compared to non-ML baseline DREAMPlace [27] and by 11% compared to SoTA machine learning based architecture RoutePlacer [16], while maintaining routed wirelength and place-and-route runtime. These results (see Appendix B, Table 16) demonstrate that **TopoGeoNet** can also translates into practical routability improvements, see more details of how we integrate in Appendix E.

Terrain SPD Problem. We evaluate using *Mean Relative Error* (MRE, lower better), *Accuracy* (higher better) defined as percentage of queries with relative error $< 2\%$, and inference run-time per endpoint pair (in milliseconds, lower better). Results are shown in Table 2. TopoGeoNet achieves the best performance on Norway and Holland, and competitive results on Philadelphia. The improvements are most significant on the challenging Norway dataset, which has unsmooth terrain landscape and larger size (see visualization of each terrain in Figure 4 in Appendix B), demonstrating the effectiveness of U-Net for capturing global geometric context and long-range interactions. For more details of dataset and model, see Appendix B and Appendix C.1.

Table 1: Average performance of different models for congestion prediction across 12 chip designs.

Model	***NRMS*↓**	***SSIM*↑**	R^2↑	**Run-time (s)↓**
pix2pixHD [29]	0.101	0.756	0.762	**0.026**
SMPNN [15]	0.102	0.890	0.821	0.097
RouteGNN [16]	0.074	0.948	0.898	0.221
DE-HNN[17]	0.098	0.918	0.849	0.189
TopoGeoNet	**0.066**	**0.963**	**0.919**	0.168

Table 2: Model Performances on Norway (4M nodes), Holland (1M nodes), and Philadelphia (1M nodes) terrain SPD datasets.

Model	***MRE* %↓**			***Accuracy*%↑**			***Run-time* ms↓**		
	Norway	Holland	Phil	Norway	Holland	Phil	Norway	Holland	Phil
GAT [24]	0.94	0.21	**0.11**	90.5	99.6	**99.9**	**0.012**	**0.008**	**0.008**
Coarse-GAT [30]	1.05	2.06	2.07	89.2	65.1	30.1	**0.012**	**0.008**	**0.008**
SMPNN [15]	0.61	0.20	0.22	95.6	99.8	99.7	0.014	0.009	0.011
TopoGeoNet	**0.29**	**0.09**	0.16	**99.4**	**99.9**	99.8	0.018	0.015	0.016

4 Conclusion

In this work, we introduce **TopoGeoNet**, a heterogeneous Neural Network framework that unifies graph neural network (GNN) message passing with convolutional U-Net processing over spatial grids. Through a bi-directional node–grid communication mechanism from U-Net, **TopoGeoNet** enriches node embeddings with global geometric context before local message passing, thereby overcoming the limitations of static virtual nodes and improving the ability to capture long-range interaction. Our experiments demonstrate that **TopoGeoNet** achieves state-of-the-art performance on large-scale tasks: chip congestion prediction on circuits with up to 1M nodes, and shortest-path estimation on terrain graphs with up to 4M nodes. When integrated into existing chip design tools [27, 28], **TopoGeoNet** provides actionable congestion feedback that leads to measurable improvements in downstream placement quality. Overall, our results show that explicitly decoupling geometric aggregation from topological propagation within a single end-to-end framework yields both accuracy and scalability, suggesting a promising direction for future research in scalable spatial graph learning.

Limitations. While **TopoGeoNet** demonstrates strong scalability and accuracy, several limitations remain. (i) The current instantiation is not *E(n)*-equivariant, since we employ vanilla MPNN and U-Net components without group-convolution constraints; extending **TopoGeoNet** with equivariant variants is an important future direction, see more discussion on this in Appendix D. (ii) Despite linear scaling in $|V|$ and $|E|$, the memory footprint becomes non-trivial on extremely large terrains ($>$ 4M nodes), where grid tensors dominate GPU usage. (iii) The model explicitly depends on node coordinates to construct spatial grids, and thus assumes reliable geometric embeddings. (iv) Finally, the benefits of the Node–Grid–Node coupling are most pronounced when long-range geometric dependencies are present; on small or purely topological graphs, simpler MPNNs may suffice.

References

[1] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Int. Conf. on Machine Learning*, ICML, page 1263–1272, 2017. 1

[2] Stefanie Jegelka. Theory of graph neural networks: Representation and learning. *Int. Congress of Mathematicians, ICM*, 2022. 1

[3] Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11604. URL https://cdn.aaai.org/ojs/11604/11604-13-15132-1-2-20201228.pdf. 1, 16

[4] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/pdf?id=S1ldO2EFPr. 1, 16

[5] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=i80OPhOCVH2. 1, 16

[6] Jake Topping, Francesco Di Giovanni, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. Understanding over-squashing and bottlenecks on graphs via curvature. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=7UmjRGzp-A. 1, 16

[7] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *Proceedings of ICML*. PMLR, 2021. URL https://proceedings.mlr.press/v139/satorras21a/satorras21a.pdf. 1

[8] Weitao Du, He Zhang, Yuanqi Du, Qi Meng, Wei Chen, Nanning Zheng, Bin Shao, and Tie-Yan Liu. SE(3) equivariant graph neural networks with complete local frames. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5583–5608. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/du22e.html. 1

[9] Hongbin Pei, Bingzhe Wei, Kevin Chen-Chuan Chang, Yu Lei, and Bo Yang. Geom-gcn: Geometric graph convolutional networks. In *International Conference on Learning Representations (ICLR)*, 2020. 1

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://papers.neurips.cc/paper/7181-attention-is-all-you-need.pdf. 1

[11] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/f1c1592588411002af340cbaedd6fc33-Abstract.html. 4

[12] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *arXiv:2205.12454*, 2022. URL https://arxiv.org/abs/2205.12454. 1

[13] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu. Routenet: Routability prediction for mixed-size designs using convolutional neural network. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2018. 1

[14] Hao Li, Xue Liu, Yixuan Wen, Zhenyu Li, Xintong Wang, and Guan Wang. A lightweight inception boosted u-net neural network for routability prediction and drc hotspot detection. *arXiv:2402.10937*, 2024. URL https://arxiv.org/pdf/2402.10937. 1

[15] Haitz Sáez de Ocáriz Bordé, Artem Lukoianov, Anastasis Kratsios, Michael Bronstein, and Xiaowen Dong. Scalable message passing neural networks: No need for attention in large graph representation learning. *arXiv preprint arXiv:2411.00835*, 2024. 2, 4, 5, 14, 15

[16] Yunbo Hou, Haoran Ye, Yingxue Zhang, Siyuan Xu, and Guojie Song. RoutePlacer: An End-to-End Routability-Aware Placer with Graph Neural Network. *30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, page 1085–1095, 2024. 2, 4, 5, 11, 14, 15, 17, 21, 22

[17] Zhishang Luo, Truong Son Hy, Puoya Tabaghi, Donghyeon Koh, Michaël Defferrard, Elahe Rezaei, Ryan Carey, Rhett Davis, Rajeev Jain, and Yusu Wang. De-hnn: An effective neural model for circuit netlist representation. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238 of *Proceedings of Machine Learning Research*. PMLR, 2024. 2, 3, 4, 5, 14, 15

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241. Springer, 2015. doi: 10.1007/978-3-319-24574-4_28. 3

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2117–2125, 2017. doi: 10.1109/CVPR.2017.106. 3

[20] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *European Conference on Computer Vision (ECCV)*, pages 801–818, 2018. doi: 10.1007/978-3-030-01234-2_49. 3

[21] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Strategies for pre-training graph neural networks. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJlWWJSFDH. 3

[22] Chen Cai, Yizhou Luo, Di He, Fei Nie, Xiao Li, and Liwei Wang. Mpnn goes transformer: Learning on large graphs via graph transformers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. doi: 10.1109/TPAMI.2023.3235301.

[23] Jaemin Hwang, Sejun Oh, Namkyeong Park, and Eunho Yang. Augmenting graph neural networks with virtual nodes for long-range reasoning. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2206.08164. 3

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *Int. Conf. on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ. 3, 4, 5, 15

[25] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Yixiao Duan, and Yiran Chen. Net2: A graph attention network method customized for pre-placement net length estimation. In *Asia and South Pacific Design Automation Conference*, ASPDAC, page 671–677, 2021. ISBN 9781450379991. doi: 10.1145/3394885.3431562. URL https://doi.org/10.1145/3394885.3431562. 4

[26] Stephen Yang, Aman Gayasen, Chandra Mulpuri, Sainath Reddy, and Rajat Aggarwal. Routability-driven FPGA placement contest. *Proc. of the International Symposium on Physical Design (ISPD)*, pages 139–143, 2016. 4

[27] Rachel Selina Rajarathnam, Mohamed Baker Alawieh, Zixuan Jiang, Mahesh Iyer, and David Z. Pan. DREAMPlaceFPGA: An Open-Source Analytical Placer for Large Scale Heterogeneous FPGAs using Deep-Learning Toolkit. *27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 300–306, 2022. 4, 5, 11, 14, 19, 20, 21

[28] ISPD'2016 Contest. Xilinx vivado v2015.4, 2016. URL https://www.ispd.cc/contests/16/Flow.txt. 4, 5, 11

[29] Mohamed Baker Alawieh, Wuxi Li, Yibo Lin, Love Singhal, Mahesh A. Iyer, and David Z. Pan. High-Definition Routing Congestion Prediction for Large-Scale FPGAs. *25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 26–31, 2020. 4, 5, 11, 14, 15, 21, 22

[30] Samantha Chen, Pankaj K Agarwal, and Yusu Wang. De-coupled neurogf for shortest path distance approximations on large terrain graphs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025. URL https://openreview.net/forum?id=PfyyEeVzQW. Poster; cite OpenReview until PMLR volume/pages are available. 4, 5, 14, 15

[31] Ladislav Rampášek, Mikhail Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.12454. 4, 14

[32] S. Yang, Z. Yang, D. Li, Y. Zhang, Z. Zhang, G. Song, and J. Hao. Versatile multi-stage graph neural network for circuit representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022. 11

[33] U.S. Geological Survey. Usgs digital elevation models (dem). https://www.usgs.gov/the-national-map-data-delivery, 2017. U.S. Geological Survey, The National Map, 3D Elevation Program. 11

[34] Kartverket (Norwegian Mapping Authority). National elevation model (dem) of norway. https://www.geonorge.no/, 2025. 11

[35] Benyamin Ghojogh, Ali Ghodsi, Fakhri Karray, and Mark Crowley. Spectral, probabilistic, and deep metric learning: Tutorial and survey. *arXiv preprint arXiv:2201.09267*, 2022. 11

[36] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *Int. Conf. on Learning Representations*, 2019. 12

[37] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. 14

[38] Raghunathan Ramakrishnan, Pavlo O. Dral, Matthias Rupp, and O. Anatole von Lilienfeld. Quantum chemistry structures and properties of 134k molecules. *Scientific Data*, 1:140022, 2014. 14, 18

[39] O. Ronneberger, P.Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015. 15

[40] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1263–1272, 2017. 19

[41] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E(n) equivariant graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 9323–9332, 2021. 19

[42] Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 2990–2999, 2016. 19

[43] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Harmonic networks: Deep translation and rotation equivariance. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5028–5037, 2017. 19

[44] Gengjie Chen, Chak-Wa Pui, Wing-Kai Chow, Ka-Chun Lam, Jian Kuang, Evangeline F. Y. Young, and Bei Yu. RippleFPGA: Routability-Driven Simultaneous Packing and Placement for Modern FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 37(10):2022–2035, 2018. 19

[45] Wuxi Li, Yibo Lin, and David Z Pan. elfPlace: Electrostatics-based Placement for Large-Scale Heterogeneous FPGAs. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8, 2019.

[46] Wuxi Li, Shounak Dhar, and David Z Pan. UTPlaceF: A routability-driven FPGA placer with physical and congestion aware packing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 37(4):869–882, 2017.

[47] Ziad Abuowaimer, Dani Maarouf, Timothy Martin, Jeremy Foxcroft, Gary Gréwal, Shawki Areibi, and Anthony Vannelli. GPlace3. 0: Routability-driven analytic placer for UltraScale FPGA architectures. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 23(5):1–33, 2018. 19

[48] Peter Spindler and Frank M. Johannes. Fast and accurate routing demand estimation for efficient routability-driven placement. In *2007 Design, Automation & Test in Europe Conference & Exhibition*, pages 1–6, 2007. doi: 10.1109/DATE.2007.364463. 20, 21

[49] MLCAD. IEEE/ACM MLCAD 2023 FPGA Macro-Placement Contest, 2023. URL https://github.com/TILOS-AI-Institute/MLCAD-2023-FPGA-Macro-Placement-Contest. 21

Appendix

A Node–Grid Communication

Algorithm 1 defines the two core procedures for exchanging information between graph nodes v and grid cells $grid$ in **TopoGeoNet**. The overall goal is to construct grid embeddings $\mathcal{H}^{\ell}(grid)$ that summarize spatial neighborhoods of nodes and then redistribute these enriched features back to the original nodes.

UpdateGrid. Given node embeddings $\{\mathcal{H}^{\ell}(v)\}$ and their coordinates $\{\mathbf{p}_v\}$, we partition the spatial domain Ω into $W \times H$ axis-aligned cells B with strides (Δ_x, Δ_y). Each cell $B_{i,j}$ collects the subset of nodes whose coordinates fall into its bounding box. Their embeddings are aggregated with a permutation-invariant operator $\oplus$ over feature functions $f \in \mathcal{F}$ (e.g., sum, mean, max). The aggregated vector is then passed through a trainable $\mathrm{MLP}_{\text{grid}}$, producing a grid embedding $\mathcal{G}^{\ell}(\text{grid})$. Empty cells default to a zero vector. This procedure projects node embeddings onto a regular grid lattice that can be processed efficiently by a U-Net.

GridAssign. After U-Net processing of the grid embeddings, enriched features need to be propagated back to local graph nodes. For each node v with position (x_v, y_v), we compute its grid index (i, j) by integer division with strides (Δ_x, Δ_y). The updated node embedding $\mathcal{H}^{\ell}_{\text{from\_grid}}(v)$ is assigned as the grid embedding $\mathcal{G}^{\ell}(\text{grid})$ at that location. In this way, each node inherits contextual information gathered from its spatial neighborhood and transformed through the multi-scale U-Net.

Overall. Together, UPDATEGRID and GRIDASSIGN enable bi-directional communication between the graph domain and the grid domain: node embeddings are pooled into grid cells for multi-scale geometric encoding, and the enriched grid embeddings are redistributed back to nodes for topological message passing. This mechanism is crucial for the seamless local-global information flow for large-scale spatial graph learning.

Algorithm 1 Node–Grid Communication (Axis-Aligned Bounding Boxes)

Domain: Bounding box $\Omega = [x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$
Grid spec: Divide into W columns and H rows
Strides: $\Delta_x = \frac{x_{\max} - x_{\min}}{W}$, $\Delta_y = \frac{y_{\max} - y_{\min}}{H}$
function UPDATEGRID(ℓ, $\{\mathcal{H}^{\ell}(v)\}$, $\{\mathbf{p}_v = (x_v, y_v)\}$, W, H)
    $\mathcal{G}^{\ell}(\text{grid}) \leftarrow \emptyset$
    for $i = 0$ **to** $W-1$ **do**
        for $j = 0$ **to** $H-1$ **do**
            $B_{i,j} \leftarrow [x_{\min}+i\Delta_x,\ x_{\min}+(i+1)\Delta_x) \times [y_{\min}+j\Delta_y,\ y_{\min}+(j+1)\Delta_y)$
            $V_{i,j} \leftarrow \{\, v : (x_v, y_v) \in B_{i,j} \,\}$
            if $V_{i,j} = \emptyset$ **then**
                agg $\leftarrow \mathbf{0}$
            else
                agg $\leftarrow \bigoplus_{f \in \mathcal{F}} f\big(\{\mathcal{H}^{\ell}(v) : v \in V_{i,j}\}\big)$
            end if
            $\mathcal{G}^{\ell}(\text{grid})[i, j] \leftarrow \mathrm{MLP}_{\text{grid}}(\text{agg})$
        end for
    end for
    return $\mathcal{G}^{\ell}(\text{grid})$
end function

function GRIDASSIGN(ℓ, $\{\mathbf{p}_v = (x_v, y_v)\}$, $\mathcal{G}^{\ell}(\text{grid})$, W, H)
    for all $v \in \mathcal{V}$ **do**
        $i \leftarrow \lfloor (x_v - x_{\min})/\Delta_x \rfloor$
        $j \leftarrow \lfloor (y_v - y_{\min})/\Delta_y \rfloor$
        $\mathcal{H}^{\ell}_{\text{from\_grid}}(v) \leftarrow \mathcal{G}^{\ell}(\text{grid})[i, j]$
    end for
    return $\{\mathcal{H}^{\ell}_{\text{from\_grid}}(v)\}$
end function

B Dataset Details

Code and Data Availability: The code and dataset used in this work are all available at our official github repo at https://github.com/luckyjackluo/TopoGeoNet.In particular, we provide extended details about our FPGA chip congestion dataset. Beyond supporting our experiments, this dataset represents a contribution to the spatial graph learning community: large-scale spatial graph benchmarks with confirmed long-range interactions are scarce, and our release offers both a reproducible data generation pipeline and a complete downstream task evaluation framework. We hope this resource will facilitate future research on scalable learning for spatial graphs.

B.1 Training and Test Dataset Generation.

B.1.1 Chip Congestion Dataset.

We generate 50 distinct legalized placement solutions per design using DREAMPlaceFPGA [27], as the Vivado [28] router requires legal placements, see Figure 2. To vary routability, we sweep the density target from 0.5 to 1.0 in 0.1 increments, producing five configurations, lower densities typically reduce congestion but worsen wirelength. For each configuration, we apply 10 random seeds to vary net weighting during wirelength optimization, resulting in $5 \times 10 = 50$ placements for each design. For the *Net-based wirelength regression* task, the golden HPWL of each net is reported by DREAMPlaceFPGA. For the *Map-based congestion regression* task, we run Vivado in non-timing-driven mode and extract PIP utilization from INT tiles using the Tcl command *"get_pips"*, collecting 50 interconnect utilization maps per design. In order to test the cross-design congestion prediction (the model will predict on the design that's not in the training dataset), we do cross-validation training on the original full dataset with 12 chip designs (graphs). Each time we use 11 designs as training dataset and use the one design left to test, following previous works practices [16, 29, 32].

Figure 2: Generate training data.

B.1.2 Terrain SPD Dataset.

The digital elevation models for Holland and Philadelphia were obtained from the US geological survey [33] and the Norway digital elevation model were obtained from The Norwegian Mapping Authority (Kartverket) [34]. Holland, Philadelphia, and Norway each come at 1.52, 3, and 10 meters of grid resolution, respectively. To generate the train sets, we sample 1,000 random sources points per terrain and then select 500 targets per source to create a train dataset with 50,000 pairs of source and target nodes and the shortest path distance between them. For the test sets, we again sample 1,000 random source points per terrain, but this time pair each source with 500,000 target nodes. This test set size ensures that all types of source–target configurations are evaluated, including challenging cases such as source or target points located on mountain ridges. In total, we evaluate our models on 500M pairs of shortest path distances per terrain.

Siamese Network. In machine learning, learning shortest path distance queries are related to metric learning, where one trains a model to learn a distance function between over some metric space X from a set of pairwise distances. Oftentimes, metric is learning is done via a *Siamese network* where given some $x, y \in X$ and a desired distance function $d(x, y)$, we train a neural network ϕ such that $\|\phi(x) - \phi(y)\|_p \approx d(x, y)$ [35]. In the case of terrain shortest paths, we take x, y to be two vertices in the terrain graph and $d(x, y)$ to be the shortest path distance between them. In our training set-up, given some instantiation of ϕ (with TopoGeoNet, GAT, or SMPNN), we estimate the terrain shortest path distance between two vertices x and y as $\|\phi(x) - \phi(y)\|_1$. Note that ϕ is then trained using the mean squared error between the neural network approximation and the true shortest path distance.

B.2 ML Dataset Convert.

We then collect all the raw generated data and convert them into ML-friendly format as pytorch-geometric [36] (heterogeneous) graph data objects. The corresponding feature name and the description of each feature we used in **chip congestion dataset** can be found in Table 3, and similar information for **terrain SPD dataset** can be found in Table 4.

Table 3: Chip Congestion Dataset: List of cell (v), net (e), and grid (g) features. Grid features at initialization are computed as the sum of all node features within the same spatial grid cell.

Feature Name	Description
Node Features ($v \in V$)	
`node_type`	Discrete ID indicating cell type
`in_degree`	# nodes driving v through nets
`out_degree`	# nodes driven by v through nets
`src2net_inst_deg`	# nets connected to v when v is a source
`node_size_x`	Width of v
`node_size_y`	Height of v
`eig_vec`	Top-5 non-trivial Laplacian eigenvectors
p_v	(x, y) coordinates of v
Net(hyperedge) Features ($e \in E$)	
`src2net_net_deg`	# source nodes connected to e
`sink2net_net_deg`	# sink nodes connected to e
p_e	(x, y) coordinates of e
`edge_attr`	L_1 distance between $\mathbf{p}_v$ and $\mathbf{p}_e$
Grid Features ($g \in \mathcal{G}$)	
`grid_agg_feat`	Element-wise sum of all node features in grid cell g

Table 4: Terrain SPD Dataset: List of node (v) and net (e), and grid (g) features.

Feature Name	Description
Node Features ($v \in V$)	
p_v	(x, y, z) coordinates of v
Edge Features ($e \in E$)	
`edge_attr`	L_2 distance between $\mathbf{p}_{v_i}$ and $\mathbf{p}_{v_j}$
Grid Features ($g \in \mathcal{G}$)	
`grid_agg_feat`	Element-wise sum of all node features in grid cell g

B.3 Dataset Statistics.

See the Node and Net degree distributions of generated chip designs in Table 5 and in Figure 3. For terrain SPD dataset, for nodes at corner have *degree* $= 2$ while other nodes in graph all have *degree* $= 4$. The *Norway* terrain graph has (2000×2000) 4 million nodes, *Holland* and *Philadelphia* terrain graphs both have (1000×1000) 1 million nodes.

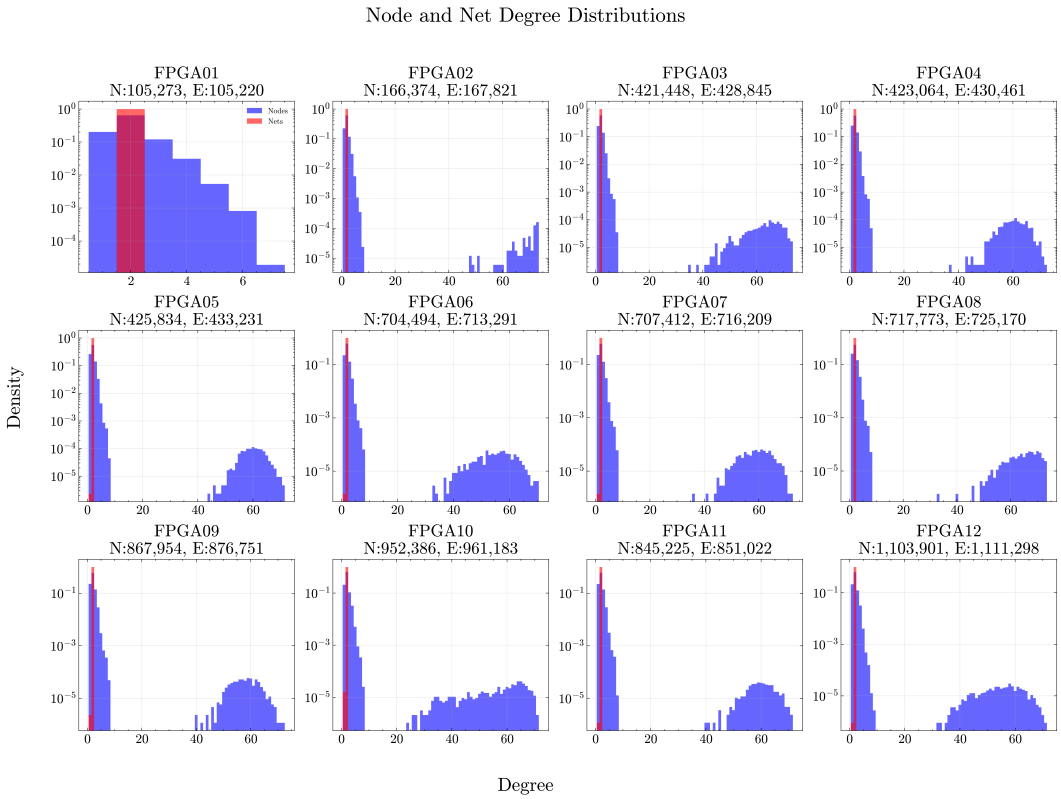

Figure 3: Degree Distribution: The degree distribution of all the FPGA Designs.

Table 5: Graph Statistics for Chip Congetion Dataset: Node and net degree statistics for each FPGA design.

Design	#Nodes	#Nets	Node Deg.		Net Deg.	
			Mean (std)	Range	Mean (std)	Range
FPGA01	105273	105220	1.99 (0.71)	0-7	2.00 (0.00)	2-2
FPGA02	166374	167821	2.01 (1.79)	0-73	2.00 (0.00)	2-2
FPGA03	421448	428845	2.03 (2.21)	0-73	2.00 (0.00)	2-2
FPGA04	423064	430461	2.03 (2.14)	0-72	2.00 (0.00)	2-2
FPGA05	425834	433231	2.03 (2.13)	0-71	2.00 (0.00)	1-2
FPGA06	704494	713291	2.02 (1.71)	0-70	2.00 (0.00)	1-2
FPGA07	707412	716209	2.02 (1.82)	0-72	2.00 (0.00)	1-2
FPGA08	717773	725170	2.02 (1.83)	0-73	2.00 (0.00)	2-2
FPGA09	867954	876751	2.02 (1.64)	0-72	2.00 (0.00)	1-2
FPGA10	952386	961183	2.01 (1.54)	0-71	2.00 (0.01)	1-2
FPGA11	845225	851022	2.04 (1.45)	0-71	2.00 (0.00)	1-2
FPGA12	1103901	1111298	2.01 (1.33)	0-71	2.00 (0.00)	1-2

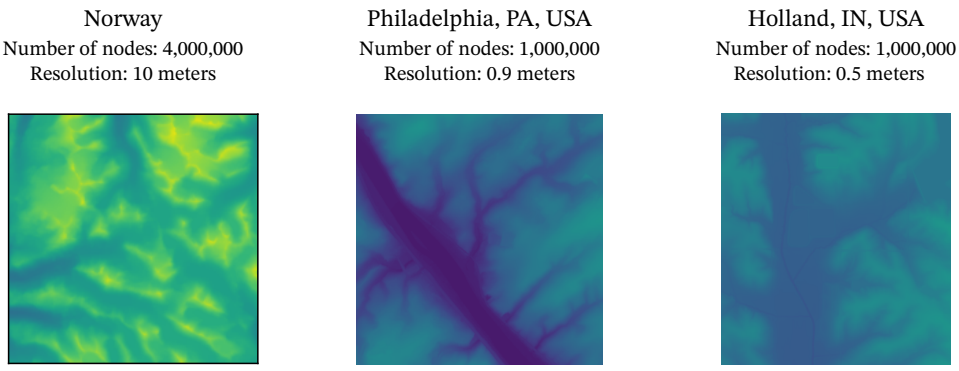

Figure 4: Terrain Maps: Comparisons of different terrain landscapes.

C Experimental Results

We show the full results for chip congestion prediction for each design(graph) in the Table 7, and the comparison visualization of congestion map prediction for one of the designs in Figure 5. We show the full results for chip congestion dataset in 7. For the terrain SPD prediction, we showed the full results in the main body of this extended abstract, and we also put the same table here as well for easier reading, see Table 6.

C.1 Experiment Setup

Our **TopoGeoNet** model is implemented in PyTorch Geometric and integrated into the open-source placer DREAMPlaceFPGA [27], which is PyTorch-based with custom C++/CUDA kernels. All experiments are conducted on a server equipped with an AMD EPYC 7462 32-core CPU and an NVIDIA RTX A100 (80 GB). Memory usage varies by dataset, and **TopoGeoNet** requires up to 52 GB GPU RAM. We attempted to train Graphomer [37] and GraphGPS [31], but even with small hidden dimensions (d=16) they could not scale to our million-node spatial graphs.

Model configuration. Unless otherwise noted, all models use four message-passing layers (L=4) and hidden dimension d=32 for the chip congestion task, and L=3 for terrain SPD. For the large *Norway* terrain, we reduce d to 16 for memory efficiency. The U-Net module follows a 4-level encoder–decoder design with residual blocks and skip connections: *Encoder:* MaxPool2d + ResidualBlock, doubling channels per level ($d \to 2d \to 4d \to 8d \to 16d$); *Decoder:* ConvTranspose2d + ResidualBlock, halving channels symmetrically. This design provides efficient multi-scale aggregation with stable gradients and complements message passing without requiring deep GNN stacks. We found L=4 to offer the best accuracy–runtime trade-off, see Table 11.

Chip Congestion Prediction. We evaluate on 12 FPGA designs from ISPD'16 benchmarks using DREAMPlaceFPGA [27]. Baselines include Pix2PixHD [29], SMPNN [15], RouteGNN [16], and DE-HNN [17]. All share identical input features and training settings for fairness. Ablations on grid resolution, pooling methods are reported in Appendix C.2.2 and Appendix C.2.2. All datasets and preprocessing scripts will be released publicly for reproducibility.

Terrain SPD Prediction. We follow the setup of Chen et al. [30] and use the same terrain graphs from *Norway*, *Holland*, and *Philadelphia* (1M–4M nodes). Metrics include *Mean Relative Error* (MRE ↓), *Accuracy* (↑; fraction of predictions with <2% relative error), and inference runtime per endpoint pair (ms ↓). We additionally test robustness on irregular terrains by randomly removing and perturbing nodes (Appendix C.3). Full architecture ablations and small-graph results on QM9 [38] are provided in Appendix C.3. Results show TopoGeoNet consistently outperforms all baselines across tasks (Tables 6–7), achieving strong scalability and generalization.

Table 6: Model Performances on Norway (4M nodes), Holland (1M nodes), and Philadelphia (1M nodes) terrain SPD datasets. (Same table as the one in main body paper)

Model	***MRE* %↓**			***Accuracy*%↑**			***Run-time* ms↓**		
	Norway	Holland	Phil	Norway	Holland	Phil	Norway	Holland	Phil
GAT [24]	0.94	0.21	**0.11**	90.5	99.6	**99.9**	**0.012**	**0.008**	**0.008**
Coarse-GAT [30]	1.05	2.06	2.07	89.2	65.1	30.1	**0.012**	**0.008**	**0.008**
SMPNN [15]	0.61	0.20	0.22	95.6	99.8	99.7	0.014	0.009	0.011
TopoGeoNet	**0.29**	**0.09**	0.16	**99.4**	**99.9**	99.8	0.018	0.015	0.016

Table 7: A comparison of accuracy between different models for congestion prediction.

Design	***NRMS*↓**					***SSIM*↑**					***R*^{2}↑**				
	pix2pixHD[29]	*SMPNN*[15]	*RouteGNN*[16]	*DE-HNN*[17]	Ours	[29]	[15]	[16]	[17]	Ours	[29]	[15]	[16]	[17]	Ours
FPGA01	0.068	0.064	0.074	0.058	**0.057**	0.913	0.892	**0.964**	0.936	**0.964**	0.899	0.840	0.928	0.899	**0.936**
FPGA02	0.100	0.102	**0.063**	0.097	0.065	0.768	0.897	**0.973**	0.886	0.971	0.839	0.890	**0.942**	0.818	**0.942**
FPGA03	0.096	0.125	0.100	**0.088**	0.098	0.846	0.907	0.917	**0.953**	0.947	0.821	0.803	0.840	**0.905**	0.848
FPGA04	0.117	0.105	0.068	0.093	**0.060**	0.721	0.921	0.966	0.951	**0.975**	0.770	0.870	0.934	0.900	**0.948**
FPGA05	0.111	0.108	0.067	0.105	**0.066**	0.767	0.846	0.972	0.888	**0.974**	0.825	0.826	**0.945**	0.791	**0.945**
FPGA06	0.129	0.116	0.073	0.094	**0.060**	0.605	0.906	0.950	0.940	**0.970**	0.623	0.813	0.897	0.878	**0.930**
FPGA07	0.146	0.126	0.085	0.094	**0.080**	0.600	0.901	0.929	0.934	**0.940**	0.516	0.773	0.863	0.873	**0.880**
FPGA08	0.045	0.059	**0.041**	0.122	**0.041**	**0.961**	0.905	**0.975**	0.864	0.970	0.931	0.854	**0.950**	0.733	**0.950**
FPGA09	0.128	0.119	0.070	0.094	**0.068**	0.531	0.880	**0.972**	0.925	**0.972**	0.779	0.773	0.944	0.859	**0.950**
FPGA10	0.110	0.107	0.084	0.102	**0.057**	0.832	0.871	0.926	0.919	**0.968**	0.718	0.795	0.861	0.798	**0.937**
FPGA11	0.102	0.108	0.074	0.105	**0.061**	0.851	0.883	0.951	0.905	**0.968**	0.792	0.872	0.907	0.815	**0.938**
FPGA12	0.112	0.111	0.108	0.108	**0.097**	0.813	0.871	0.884	0.892	**0.940**	0.734	0.759	0.782	0.801	**0.833**
Geomean	0.101	0.102	0.074	0.098	**0.066**	0.756	0.890	0.948	0.918	**0.963**	0.762	0.821	0.898	0.849	**0.919**

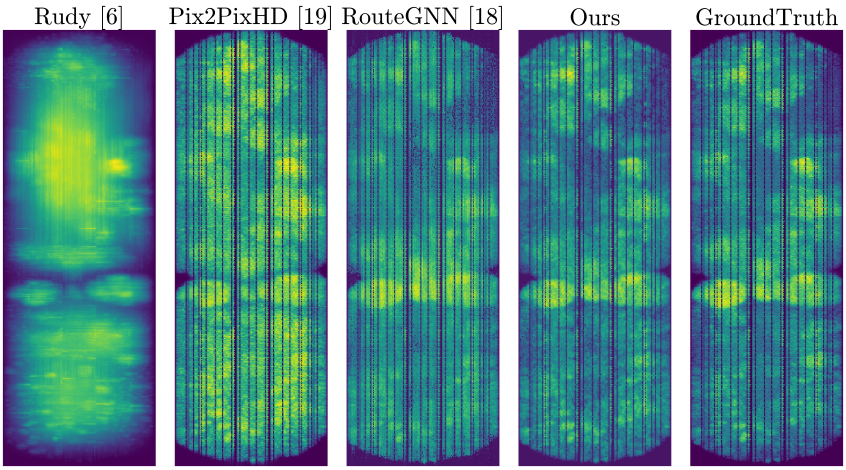

Figure 5: Congestion Maps: Comparisons of different predicted congestion maps and the ground truth, design FPGA05.

C.2 Ablation Study for TopoGeoNet

We conducted an ablation study to analyze the effects of different components used in our **TopoGeoNet**. For practicable run-time we did not run ablation study on all datasets we have, but select four chip design graphs with scales range from 100k nodes to 1M nodes to form a smaller dataset to test all variants of models, see Table 10.

C.2.1 Different Variants of TopoGeoNet

We use DE-HNN [17] as base MPNN without any additional components as our baseline and build different variants by adding the important components **Fourier Positional Encoder**, **Grid Nodes** and **U-Net** [39] to the baseline, and gradually build the final *full* **TopoGeoNet** which has all components.

Base DE-HNN. The base DE-HNN model contains only the Message Passing mechanism with the node features $m(v)$, net features $M(e)$, and FPGA design netlist $H(V, E)$ as input. The $\mathbf{p}_v$ and $\mathbf{p}_e$ are directly input with the other node/net features without any special encoding process. To enable the model to predict the congestion map, at the readout layer we still map the node embeddings to the grid embeddings and process the final MLP output layer. The difference between this baseline and

the DE-HNN + Grid version is that there are no grid nodes and corresponding communications in the intermediate layers.

Base DE-HNN + Fourier Positional Encoder. To enrich raw spatial coordinates with high-frequency information, we augment the base DE-HNN with a fixed *random Fourier positional encoder* applied to *node coordinates only*. Given d-dimensional coordinate vector $\mathbf{p}_v \in \mathbb{R}^d$, we pre-sample a Gaussian projection matrix $B \in \mathbb{R}^{d\times(m/2)}$ once at initialization with $B_{ij} \sim \mathcal{N}(0, \sigma^2)$, where m is the target embedding dimension and σ is a scaling factor. The encoded representation is computed as

$$\gamma(\mathbf{p}_v) = [\sin(2\pi\mathbf{p}_v\mathbf{B}),\ \cos(2\pi\mathbf{p}_v\mathbf{B})] \in \mathbb{R}^m,$$

We replace raw coordinates by their encoded versions in the node features and concatenate:

$$\widetilde{m}(v) \ = \ \big[\, m(v) \parallel \gamma(\mathbf{p}_v) \,\big].$$

An input MLP maps $\widetilde{m}(v)$ to the initial node embeddings, after which the pipeline is identical to the base DE-HNN: standard message passing over $H(V, E)$ with *no* grid nodes or U-Net modules in intermediate layers, and a readout that projects node embeddings to grid embeddings followed by an MLP to predict the congestion map. Random Fourier features inject multi-frequency sinusoidal bases that improve coordinate expressivity with negligible overhead; (m, σ) are selected by validation, and B is sampled once and kept fixed for reproducibility.

DE-HNN (+ Fourier Encoder) + Grid. Starting from this variant, we will always include the Fourier Positional Encoder but we will no longer explicitly mention that. The DE-HNN + Grid version has Grid Nodes grid at all layers that allow communication between nodes (instances) V and the corresponding grids through functions UPDATEGRID and GRIDASSIGN. This communication allows the model to capture spatial-based intermediate grid embeddings $\mathcal{G}^\ell$ at each layer ℓ and pass that information to the node embeddings $\mathcal{H}^\ell(v)$. We also include the ablation study on grid sizes in C.2.2.

TopoGeoNet: DE-HNN (+ Fourier Encoder) + Grid + U-Net. The full version model **TopoGeoNet** has all components. Compared to the DE-HNN + Grid version, **TopoGeoNet** has two additional U-Net models at the first layer and at the last layer. Each U-Net model performs hierarchical encoding and decoding for the grid embeddings $\mathcal{G}^\ell$, and this process allows the model to capture spatial geometric information from different resolutions/scales. Additionally, **Message Passing** is known to have issues such as *over-smoothing* [3, 4] and *over-squashing* [5, 6]. The U-Net added to the first layer of DE-HNN also serves as a spatial hierarchical virtual node structure that allows long-range global interactions. We show that compared to other variants, **TopoGeoNet** achieves the best prediction accuracy. See Section C.2.2. We call this as our *full* **TopoGeoNet**.

C.2.2 Experimental Results for Ablation Study

We selected four representative FPGA designs (FPGA01, FPGA04, FPGA05, FPGA12) to conduct systematic ablations across all variants. These designs span diverse complexity profiles (netlist size, placement density, and congestion patterns). See the detailed results in Table 10.

Effect of Fourier positional encoder. Augmenting DE-HNN with the (fixed) random Fourier encoder on node coordinates yields consistent gains over the base model. On the geometric mean across the four designs ("Geo." row), NRMS drops by 5.7%, SSIM increases by 1.4%, and R^2 improves by 2.5%. Per-design, the largest NRMS reductions occur on *FPGA05* ($\downarrow$ 8.1%) and *FPGA12* ($\downarrow$ 5.8%), with smaller but consistent improvements on *FPGA01* ($\downarrow$ 5.6%) and *FPGA04* ($\downarrow$ 1.8%); SSIM improves slightly on all four (up to $\uparrow$ 2.3% on *FPGA12*), and R^2 gains are most pronounced on *FPGA12* ($\uparrow$ 5.2%).

Effect of Grid Nodes. Introducing Grid Nodes into DE-HNN leads to substantially improvements than Fourier encoding alone. Relative to the base DE-HNN, NRMS decreases by 15.2%, SSIM rises by 3.7%, and R^2 increases by 6.6% on average (Geo.). Compared to the Fourier variant, Grid Nodes provide an additional 9.7% NRMS reduction, a further 2.2% SSIM increase, and a 4.0% gain in R^2. Improvements are strongest on *FPGA01* and *FPGA05*, while *FPGA04* shows more modest gains. As we discussed in the main body of this extended abstract, grids $\mathcal{G}$, as local virtual nodes can help capture limited long-range interactions and learn local geometric context. The results both from this

variant and also from *RouteGNN* [16] suggest adding grids are helpful for spatial graph learning besides manual designed geometric positional encoding (e.g. Fourier Positional Encoding).

Grid Resolution Study. We analyze the effect of grid resolution (W, H) on model accuracy and efficiency. In TopoGeoNet, the grid resolution determines how finely the spatial field is discretized for U-Net aggregation, controlling the balance between geometric fidelity and computational cost. Table 8 compares three settings on four representative FPGA designs: (120×42), (240×84), and (480×168). Coarser grids substantially degrade performance in both structural similarity (SSIM) and R^2, as they lose local geometric details; finer grids recover these patterns with minimal runtime increase due to U-Net's high GPU parallelism. Empirically, the (480×168) grid achieves the optimal trade-off between accuracy and runtime and is used as the default configuration for all chip-design experiments. This result validates that TopoGeoNet benefits from denser geometric aggregation without suffering significant computational overhead.

Table 8: Ablation on grid resolution for TopoGeoNet. Finer grids yield higher accuracy with minimal runtime overhead.

Grid Size	NRMSE ↓	SSIM ↑	R^2 ↑	Runtime (s) ↓
120×42	0.112	0.860	0.756	0.147
240×84	0.092	0.894	0.818	0.152
480×168	**0.068**	**0.963**	**0.914**	0.164

Full TopoGeoNet (Fourier Encoder + Grid + U-Net). Adding U-Net at input/output on top of Grid Nodes delivers the best results across all configurations. Against base DE-HNN, **TopoGeoNet** achieves 31.3% lower NRMS, 9.2% higher SSIM, and 14.4% higher in R^2 on Geo. Relative to the Grid-only variant, **TopoGeoNet** further improves NRMS by 19.1%, SSIM by 5.2%, and R^2 by 7.3%. Compared directly to the Fourier variant, the gains are 26.9% NRMS reduction, 7.6% SSIM increase, and 11.6% R^2 boost, highlighting that U-Net as learnable hierarchical spatial processing provides substantial benefits in multi-scale geometric context learning and in capturing long-range interactions.

Pooling Baseline Comparison. We further evaluate TopoGeoNet against hierarchical pooling variants to analyze the benefits of our grid-based aggregation. Pooling methods such as Metis or K-means rely on precomputed partitions of the graph and static virtual nodes, whereas TopoGeoNet's learnable U-Net performs adaptive, data-driven aggregation on a continuous grid domain. Table 9 reports quantitative results on four representative FPGA designs (FPGA01, FPGA04, FPGA05, FPGA12). While clustering-based pooling improves scalability over naïve message passing, it incurs significant preprocessing cost and cannot adapt to geometric features. Our grid-based U-Net not only achieves the highest accuracy across all metrics but also maintains competitive runtime by completely eliminating partitioning overhead. These results confirm that dynamic, grid-level convolutional hierarchies provide both greater flexibility and efficiency compared to static partition-based pooling.

Table 9: Comparison of pooling baselines. Runtime decomposed into (partitioning + model inference).

Model	NRMSE ↓	SSIM ↑	R^2 ↑	Runtime (s) ↓
Metis + Pooling (DE-HNN)	0.091	0.917	0.848	0.187 (0.071 + 0.116)
K-means + Pooling	0.090	0.920	0.851	0.685 (0.580 + 0.105)
Grids + Pooling	0.081	0.932	0.865	0.131 (0.021 + 0.110)
Grids + U-Net (TopoGeoNet)	**0.068**	**0.963**	**0.914**	0.164 (0.020 + 0.144)

Ablation on message-passing depth. We further varied the number of MPNN layers L. Results show that $L = 4$ achieves the best empirical balance between expressivity and efficiency. Reducing L to 3 harms both NRMSE and SSIM significantly, indicating insufficient topological propagation. Increasing L to 5 offers no gain—likely due to redundant long-range modeling already captured by the U-Net's hierarchical aggregation, see results in Table 11.

C.3 Experimental Results for Additional Tasks

Irregular Terrain Robustness. To further assess the robustness of TopoGeoNet under irregular topologies, we constructed a new terrain graph from the Norway Digital Elevation dataset by

Table 10: Ablation Study: Performances of **TopoGeoNet** with/without components. Variants *DE-HNN + Grid* and *DE-HNN + Grid + U-Net* (**TopoGeoNet**) both include Fourier positional encoder.

NRMS (↓)				
Design	base DE-HNN	DE-HNN+Fourier Enc.	DE-HNN+Grid	**TopoGeoNet**
FPGA01	0.074	0.070 (↓5.6%)	0.069 (↓6.7%)	0.057 (↓23.0%)
FPGA04	0.094	0.092 (↓1.8%)	0.086 (↓8.5%)	0.060 (↓36.2%)
FPGA05	0.125	0.115 (↓8.1%)	0.104 (↓16.8%)	0.066 (↓47.2%)
FPGA12	0.109	0.103 (↓5.8%)	0.100 (↓8.3%)	0.097 (↓11.0%)
Geo	0.099	0.093 (↓5.7%)	0.084 (↓15.2%)	0.068 (↓31.3%)
SSIM (↑)				
Design	base DE-HNN	DE-HNN+Fourier Enc.	DE-HNN+Grid	**TopoGeoNet**
FPGA01	0.862	0.876 (↑1.6%)	0.909 (↑5.5%)	0.964 (↑11.7%)
FPGA04	0.948	0.957 (↑1.0%)	0.958 (↑1.1%)	0.975 (↑2.8%)
FPGA05	0.843	0.852 (↑1.1%)	0.896 (↑6.3%)	0.974 (↑15.5%)
FPGA12	0.877	0.897 (↑2.3%)	0.899 (↑2.5%)	0.940 (↑7.2%)
Geo	0.882	0.895 (↑1.4%)	0.915 (↑3.7%)	0.963 (↑9.2%)
$\mathbf{R^2}$ (↑)				
Design	base DE-HNN	DE-HNN+Fourier Enc.	DE-HNN+Grid	**TopoGeoNet**
FPGA01	0.836	0.855 (↑2.3%)	0.903 (↑8.0%)	0.936 (↑12.0%)
FPGA04	0.899	0.908 (↑1.0%)	0.916 (↑1.9%)	0.948 (↑5.5%)
FPGA05	0.706	0.715 (↑1.2%)	0.794 (↑12.5%)	0.945 (↑33.9%)
FPGA12	0.769	0.809 (↑5.2%)	0.804 (↑4.6%)	0.833 (↑8.3%)
Geo	0.799	0.819 (↑2.5%)	0.852 (↑6.6%)	0.914 (↑14.4%)

Table 11: Ablation on the number of message-passing (MPNN) layers (L).

Model	**NRMSE** ↓	**SSIM** ↑	R^2 ↑	**Runtime (s)** ↓
3-layer	0.076	0.948	0.886	**0.115**
4-layer (default)	**0.068**	**0.963**	**0.914**	0.164
5-layer	0.069	0.961	0.913	0.192

randomly dropping nodes and perturbing their (x, y) coordinates by up to 30% of the grid spacing. This process destroys grid regularity while preserving the overall geometric structure, producing a highly irregular spatial graph with lower mean degree and higher variance (Table 12). We evaluated GAT, SMPNN, and our TopoGeoNet on this dataset using identical training settings (L=3, d=64, grid size 250×250). As shown in Table 13, TopoGeoNet maintains superior accuracy across all metrics, achieving over twofold lower mean squared error compared to SMPNN. These results highlight that TopoGeoNet's grid-based geometric aggregation generalizes well even when spatial regularity is disrupted, confirming its adaptability to continuous and non-uniform node distributions.

Table 12: Graph statistics before and after destroying grid regularity.

Graph Type	**Nodes**	**Edges**	**Mean Deg.**	**Median**	**Std**	**Min**	**Max**
Regular Grid (1000×1000)	1,000,000	1,998,000	3.996	4.0	0.063	2	4
Destructed Grid	456,249	579,486	2.54	3.0	0.89	1	4

Table 13: Performance comparison on irregular terrain graphs.

Model	**Test Loss** ↓	**MSE** ↓	**MAE** ↓	**Rel. Err.** ↓
GAT	0.0590	0.0590	0.1382	0.0259
SMPNN	0.0244	0.0244	0.0932	0.0186
TopoGeoNet	**0.0113**	**0.0113**	**0.0607**	**0.0120**

Small-Graph (Molecular) Evaluation. Although TopoGeoNet primarily targets large-scale spatial graphs, we also evaluate its flexibility on the small-molecule benchmark QM9 [38]. Each molecule is represented as a spatial graph whose nodes correspond to atoms with 3D coordinates, and edges encode chemical bonds. We predict molecular properties using three models—GAT, SMPNN, and our TopoGeoNet (GAT backbone)—under identical training configurations. As shown in Table 14,

TopoGeoNet achieves slightly lower mean squared and mean absolute errors and a marginally higher R^2 score, indicating competitive accuracy even on small graphs. However, the improvement is modest because long-range geometric dependencies and large-scale structural hierarchies, which TopoGeoNet is designed to model, are largely absent in QM9. These findings suggest that TopoGeoNet generalizes well across graph sizes while being most beneficial on large spatial graphs with complex geometry.

Table 14: Performance on the QM9 molecular dataset. While gains are minor, TopoGeoNet maintains competitive accuracy.

Model	MSE ↓	MAE ↓	R^2 ↑
GAT	0.1731	0.2901	0.9250
SMPNN	0.1698	0.2883	0.9251
TopoGeoNet (GAT base)	**0.1695**	**0.2881**	**0.9273**

D Geometric Equivariance

Equivariance in Hybrid Graph–Grid Models. A model is said to be *geometrically equivariant* if its outputs transform predictably under spatial transformations such as rotations or translations. In TopoGeoNet, equivariance can in principle arise from both components: the message-passing network (topological propagation) and the U-Net (geometric aggregation). However, in our current implementation we use a vanilla MPNN [40] and a standard convolutional U-Net, neither of which guarantees $E(n)$-equivariance. We discuss more details below with possible improvements and future work.

Graph component. While standard MPNNs are not inherently $E(n)$-equivariant, variants such as $E(n)$-GNNs [41] and SE(3)-Transformers enforce this property by constructing messages using relative positions and rotation-invariant distances. Replacing the MPNN in TopoGeoNet with an $E(n)$-equivariant GNN would yield rotation- and translation-consistent node embeddings without altering the Node→Grid→Node interface.

Grid component. Standard U-Nets employ planar convolutions that are translation-equivariant but not rotation-equivariant. Group-equivariant convolutions [42, 43] extend CNNs to respect specific transformation groups (e.g., $p4$, $SO(2)$, or $E(2)$). Integrating such group convolutions into TopoGeoNet's U-Net could enable geometric consistency under rotations and reflections of the spatial field.

Coupling and Future Work. Achieving full geometric equivariance in TopoGeoNet requires both the node-to-grid and grid-to-node mappings to preserve coordinate transformations consistently. Future directions include (i) replacing the MPNN with an $E(n)$-equivariant variant, (ii) substituting the U-Net with a group-equivariant convolutional backbone, and (iii) enforcing transformation-consistent interpolation between graph and grid domains. We expect this line of development to produce an *Equivariant TopoGeoNet* capable of learning geometry-aware representations that generalize across rotated or reflected spatial graphs.

E DREAMPlaceFPGA Integration for Routability-driven Placement flow.

We evaluate our model in a routability-driven placer using the flow illustrated in Fig. 6. Two strategies leverage the congestion predictions introduced earlier. Both congestion-aware strategies are activated only when the placement overflow condition is met, i.e., $\max(OVFL_{LUT}, OVFL_{FF}) < 0.15$.

E.1 Config 1: Two-stage placement with a predicted congestion map

A common practice in academic FPGA placement is a two-stage area inflation procedure [27, 44–47]: an estimated congestion map is generated, and cell areas in congested regions are manually inflated to reduce placement density. This method is not differentiable and relies on the accuracy of the congestion estimation. When the overflow condition is met, LUT and FF cells in congested regions

are inflated according to

$$
\begin{aligned}
congestion_{clamp} &= f_G^{exponent}.clamp(0.5, 2) \\
increment &= \max_{(x,y)\in G_v} congestion_{clamp}(x, y) \\
node\_size\_x(v) &= node\_size\_x(v) \times \sqrt{increment} \\
node\_size\_y(v) &= node\_size\_y(v) \times \sqrt{increment}.
\end{aligned}
$$

Here, *exponent*$= 2.0$, and G_v denotes the set of grid cells overlapped by cell v. If the total cell area changes by less than 1% in an iteration, the placer move on to next legalization stage. In our flow, the model performs inference on the current placement to generate a congestion map, which replaces the heuristic-based RUDY [48] estimation in the baseline DREAMPlaceFPGA [27] while preserving the same two-stage optimization and inflation procedure.

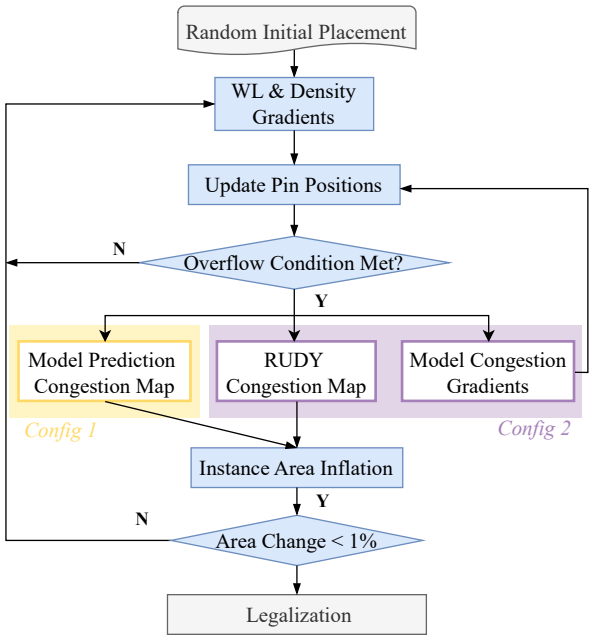

Figure 6: Routability-driven Placement flow.

E.2 Config 2: Differentiable congestion optimization through model forward and backward

Because routing congestion is considered highly non-convex and discrete, prior work rarely treats it as a relaxed, differentiable constraint that can be optimized with gradient descent. The overall placement objective is

$$\min \sum_{e\in E} W_e(x, y) + \lambda \cdot D(x, y) + \eta \cdot \mathcal{L}(x, y),$$

where W and D are wirelength and density objectives, and $\mathcal{L}$ penalizes predicted congestion. We observe that our **TopoGeoNet** model predicts congestion accurately even without RUDY [48] as inputs, implying that the architecture can implicitly learn the spatial information that RUDY [48] provides. Consequently, gradients of the congestion loss with respect to cell coordinates can be obtained directly by back-propagation, provided the computational graph is defined properly (see Fig. 1). In practice, we mark the initial cell positions $pos(v)$ with `requires_grad` *before* computing any derived quantities such as net positions $pos(e)$ or edge L_1 distances, so that gradients propagate all the way back to $pos(v)$:

$$\nabla_x \mathcal{L} \;=\; J_x\big(f_G\big)^{\mathsf{T}}\, J_{f_G}(\mathcal{L}),$$

where f_G is the predicted congestion map. We define the congestion loss as, $\mathcal{L}(x,y) = \left\| f_G(x,y) - f_{\text{target}}(x,y) \right\|_2^2$, where the target map $f_{\text{target}}(x,y)$ is specified as

$$f_{\text{target}}(x,y) = \begin{cases} 0.5 f_G(x,y), & \text{if } f_G(x,y) \in [\theta_{\min}, \theta_{\max}], \\ 0, & \text{otherwise.} \end{cases}$$

And the thresholds $\theta_{\min}$ and $\theta_{\max}$ define the clipping range corresponding to the top 75% to 99% of congestion values. We scale η by the ratio of the wirelength and congestion loss gradients:

$$\eta = c \frac{\| \nabla W_e(x,y) \|_2}{\| \nabla \mathcal{L}(x,y) \|_2},$$

where c is a constant. The congestion loss term is updated every ten iterations only when the overflow condition described above is satisfied.

E.3 Experimental Results for Routability-driven FPGA Placement

In this work, we apply **TopoGeoNet** to DREAMPlaceFPGA [27] (denoted as *DMP*) through the two configurations mentioned. However, since *pix2pixHD* [29] is not directly differentiable with respect to cell locations, it is excluded from the **Config 2** experiments. This section first defines the metrics used to evaluate placement routability, and then presents routability-driven placement results under both configurations.

E.3.1 Routability Metrics

- *Routed Wirelength*. While placement minimizes half-perimeter wirelength (HPWL), the routed wirelength reported captures extra detours required in congested regions.
- *Routability Score (ρ)*. Introduced in the MLCAD'2023 FPGA macro placement contest [49], lower scores indicate better routability. The *Routability score* [49] (see Equation (E.3.1)) comprises two components: the initial routing-congestion score Sr_i and the final routing-congestion score Sr_f. The first term Sr_i reflects the short- and global-congestion levels, L_i^{short} and L_i^{global}, reported by Vivado's initial routing phase for each of the four directions ($i \in \{\text{N}, \text{S}, \text{E}, \text{W}\}$). The second term Sr_f is measured as the number of outer iterations executed by the detailed router. We sum up the two scores as:

$$Sr_i = 1 + \sum_{i=1}^{4} (\max(0, L_i^{short} - 3)^2 + \max(0, L_i^{global} - 3)^2)$$
$$\rho = Sr_i + Sr_f$$

- *Place-and-route Runtime*. The sum of placement and routing runtimes reflects overall turnaround.

E.3.2 Routability-driven Results

Config 1: For a fair comparison, we enforce deterministic for both *pix2pixHD* [29] and **TopoGeoNet**. However, since certain Deep Graph Library operations used by *RouteGNN* [16] are inherently non-deterministic, we run the placer 10 times and report average routability metrics, and the same procedure is applied in Config 2. Recall that Config 1 uses a two-stage flow where the predicted congestion map guides instance-area inflation prior to legalization. Table 15 summarizes the results for Config 1. The baseline placer (*DMP*) is DREAMPlaceFPGA with a RUDY-based congestion map; *DMP* + [29], *DMP* + [16], and *DMP* + *Ours* denote the same placer using congestion maps from *pix2pixHD* [29], *RouteGNN* [16], and our model, respectively. None of the learned maps degrade the overall routed wirelength. On the most congested design, *FPGA05*, the learned maps avoid routing failures and reduce routed wirelength by 2%. Regarding congestion score, our model outperforms others on 7 out of 12 designs, achieving an overall 15.7% improvement over the RUDY [48] baseline. Since place-and-route runtime is largely dominated by routing, our model also leads in this metric, most notably reducing total runtime for *FPGA05* by 28%.

Config 2: Results from Config 2 in Table 16 show that applying our model's gradients to the baseline placer yields a 3.4% improvement in congestion score, compared to 1.3% from *RouteGNN* [16].

Table 15: A comparison of routability between different models applied in placer with **Config 1**.

Design	*Routed Wirelength* ($\times 10^3$)				*Congestion Score*				*Place-and-Route Runtime* (in sec)			
	DMP	DMP +[29]	DMP +[16]	DMP + Ours	DMP	DMP +[29]	DMP +[16]	DMP + Ours	DMP	DMP +[29]	DMP +[16]	DMP + Ours
FPGA01	333	**332**	333	333	**5**	**5**	6	**5**	**65**	76	77	69
FPGA02	**594**	595	599	**594**	6	**4**	**4**	5	**103**	123	124	117
FPGA03	2957	**2949**	2952	2951	**5**	7	6	7	**187**	219	212	197
FPGA04	**4907**	4917	4962	4930	7	**6**	**6**	7	**216**	240	229	222
FPGA05	9376	**9138**	9297	9188	51	57	**39**	**39**	2004	3605	**1120**	1451
FPGA06	5862	5876	**5838**	5856	8	**7**	8	**7**	**375**	416	405	397
FPGA07	**8856**	8861	8960	8989	18	20	14	**12**	652	774	490	**490**
FPGA08	7538	7533	7533	**7524**	6	**5**	6	6	306	324	316	**296**
FPGA09	**10711**	10722	10722	10712	31	27	24	**21**	721	836	**690**	701
FPGA10	6253	6220	**6188**	6221	11	12	**10**	11	530	589	499	**481**
FPGA11	**10424**	10442	10685	10758	20	22	15	**14**	479	559	493	**468**
FPGA12	6657	6659	6653	**6635**	23	17	13	**12**	570	606	**511**	517
Geomean	1.00	**0.998**	1.003	1.001	1.000	0.953	0.852	**0.843**	1.000	1.180	0.973	**0.954**

Although the backward pass adds placer runtime overhead, the overall turnaround time still improves by 2.9%. However, both models achieve smaller gains in congestion score than in Config 1, consistent with observations from [16]. While gradient-based integration appears natural in analytical placement, we observe that the learned congestion gradients sometimes misaligned with expected density gradient directions, which is from more to less congested regions, suggesting that the model might struggle to fully capture congestion dynamics in relation to placement.

Table 16: A comparison of routability between different model gradients applied in placer with **Config 2**.

Design	*Routed Wirelength* ($\times 10^3$)			*Congesiton Score*			*Place-and-Route Runtime* (in sec)		
	DMP	DMP +[16]	DMP + Ours	DMP	DMP +[16]	DMP + Ours	DMP	DMP +[16]	DMP + Ours
FPGA01	333	334	**333**	5	5	5	**65**	74	73
FPGA02	594	594	**592**	6	5	**4**	**103**	119	104
FPGA03	2957	**2948**	2954	5	5	5	187	208	**185**
FPGA04	4907	**4906**	4910	7	7	7	216	230	**208**
FPGA05	**9376**	9379	9379	**51**	55	53	2004	2167	**1912**
FPGA06	5862	5864	**5854**	**8**	**8**	9	375	403	**370**
FPGA07	**8856**	8864	8864	**18**	19	19	652	673	**609**
FPGA08	7538	7535	**7534**	6	6	**5**	306	312	**286**
FPGA09	10711	10706	**10703**	31	**30**	**30**	721	730	**686**
FPGA10	6253	6236	**6235**	**11**	13	15	530	556	**493**
FPGA11	10424	10413	**10412**	**20**	21	21	479	509	**477**
FPGA12	**6657**	6665	6663	23	17	**16**	570	541	**514**
Geomean	1.00	1.000	**0.999**	1.000	0.987	**0.966**	1.000	1.061	**0.971**