A Benchmark on Directed Graph Representation Learning in Hardware Designs

Haoyu Wang Georgia Tech
Yinan Huang Georgia Tech
Nan Wu George Washington University
Pan Li Georgia Tech

Abstract

To keep pace with the rapid advancements in design complexity within modern computing systems, directed graph representation learning (DGRL) has become crucial, particularly for encoding circuit netlists, computational graphs, and developing surrogate models for hardware performance prediction. However, DGRL remains relatively unexplored, especially in the hardware domain, mainly due to the lack of comprehensive and user-friendly benchmarks. This study presents a novel benchmark comprising five hardware design datasets and 13 prediction tasks spanning various levels of circuit abstraction. We evaluate 21 DGRL models, employing diverse graph neural networks and graph transformers (GTs) as backbones, enhanced by positional encodings (PEs) tailored for directed graphs. Our results highlight that bidirected (BI) message passing neural networks (MPNNs) and robust PEs significantly enhance model performance. Notably, the top-performing models include PE-enhanced GTs interleaved with BI-MPNN layers and BI-Graph Isomorphism Network, both surpassing baselines across the 13 tasks. Additionally, our investigation into out-of-distribution (OOD) performance emphasizes the urgent need to improve OOD generalization in DGRL models. This benchmark, implemented with a modular codebase, streamlines the evaluation of DGRL models for both hardware and ML practitioners.

1 Introduction

Directed graphs, where edges encode directional information, are widely utilized as data models in various applications, including email communication [62, 66], financial transactions [22, 41, 117], and supply chains [61, 113, 125]. Notably, hardware designs can be represented as directed graphs, such as circuit netlists [47, 124], control and data flow graphs [11, 26, 137, 144], or computational graphs [100, 150], often exhibiting unique properties. These graph structures reflect restricted connection patterns among circuit components or program operation units, with directed edges encapsulating long-range directional and logical dependencies.

Recently, employing machine learning (ML) to assess the properties of hardware designs via their directed graph representations has attracted significant attention [11, 14, 29, 45, 51, 71, 85, 100, 135]. Traditional simulation-based methods often require considerable time (hours or days) to achieve the desired accuracy in assessing design quality [27, 136, 137, 154], substantially slowing down the hardware development cycle due to repeated optimization-evaluation iterations. In contrast, ML models can serve as faster and more cost-effective surrogates for simulators, offering a balanced alternative between simulation costs and prediction accuracy [8, 15, 16, 19, 31, 59, 77, 91, 126, 134,

Emails: haoyu.wang@gatech.edu, panli@gatech.edu

Submitted to the 38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks. Do not distribute.

136]. Such an approach is promising to expedite hardware evaluation, especially given the rapid growth of design complexity in modern electronics and computing systems [111].

Despite the promising use cases, developing ML models for reliable predictions on directed graphs, particularly within hardware design loops, is still in its early stages, largely due to the lack of comprehensive and user-friendly benchmarks. Existing studies in the ML community have primarily focused on undirected graphs, utilizing Graph Neural Networks (GNNs) [63, 123, 140] or Graph Transformers (GTs) [67, 90, 101, 145]. Among the limited studies on directed graph representation learning (DGRL) [42, 118, 119, 152], most have only evaluated their models for node/link-level predictions on single graphs in domains such as web networks, or financial networks [50]. These domains exhibit very different connection patterns compared to those in hardware design. To the best of our knowledge, CODE2 in the Open Graph Benchmark (OGB) [52] is the only commonly used benchmark that may share some similarities with hardware data. However, the graphs in CODE2 are IRs of Python programs, which may not fully reflect the properties of data in hardware design loops.

Numerous DGRL models for hardware design tasks have been developed by domain experts. While promising, hardware experts tend to incorporate domain-specific insights with off-the-shelf GNNs (e.g., developing hierarchical GNNs to mimic circuit modules [29, 137] or encoding circuit fan-in and fan-out in node features [10, 102, 121]), with limited common design principles investigated in model development. In contrast, state-of-the-art (SOTA) DGRL techniques proposed by the ML community lack thorough investigation in these tasks. These techniques potentially offer a more general and effective manner of capturing data patterns that might be overlooked by domain experts.

Present Benchmark. This work addresses the aforementioned gaps by establishing a new benchmark consisting of representative hardware design tasks and extensively evaluating various DGRL techniques for these tasks. On one hand, the evaluation results facilitate the identification of commonly useful principles for DGRL in hardware design. On the other hand, the ML community can leverage this benchmark to further advance DGRL techniques.

Specifically, our benchmark collects five hardware design datasets encompassing a total of 13 prediction tasks. The data spans different levels of circuit abstraction, with graph sizes reaching up to 400+ nodes per graph across 10k+ graphs for graph-level tasks, and up to 50k+ nodes per graph for node-level tasks (see Fig. 1 and Table. 1). We also evaluate 21 DGRL models based on 8 GNN/GT backbones, combined with different message passing directions and various enhancements using positional encodings (PEs) for directed graphs [42]. PEs are vectorized representations of node positions in graphs and have been shown to improve the expressive power of GT/GNNs for undirected graphs [53, 74, 101, 127]. PEs for directed graphs are still under-explored [42], but we believe they could be beneficial for hardware design tasks that involve long-range and logical dependencies.

Our extensive evaluations provide significant insights into DGRL for hardware design tasks. Firstly, bidirected (BI) message passing neural networks (MPNNs) can substantially improve performance for both pure GNN encoders and GT encoders that incorporate MPNN layers, such as GPS [101]. Secondly, PEs, only when used stably [53, 127], can broadly enhance the performance of both GTs and GNNs. This observation contrasts with findings from undirected graph studies, particularly in molecule property prediction tasks, where even unstable uses of PEs may improve model performance [32, 67, 74, 101]. Thirdly, GTs with MPNN layers typically outperform pure GNNs on small graphs but encounter scalability issues when applied to larger graphs.

With these insights, we identify two top-performing models: GTs with BI-MPNN layers (effective for small graphs in the HLS and AMP datasets) and the BI-Graph Isomorphism Network (GIN) [140], both enhanced by stable PEs. These models outperform all baselines originally designed by hardware experts for corresponding tasks, across all 13 tasks. Notably, this work is the first to consider GTs with BI-MPNN layers and using stable PEs in DGRL, so the above two models have novel architectures essentially derived from our benchmarking effort.

Furthermore, recognizing that hardware design often encounters out-of-distribution (OOD) data in production (e.g., from synthetic to real-world [137], before and after technology mapping [134], inference on different RISC-V CPUs [51]), for each dataset we evaluate the methods data with

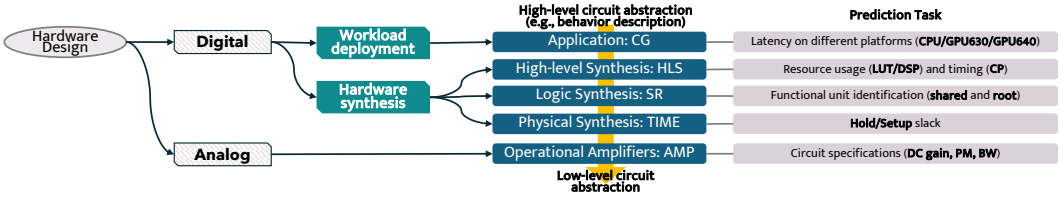

Figure 1: Coverage of Datasets/Tasks.

		High-level Synthesis (HLS) [137]	Symbolic Reasoning (SR) [134]	Pre-routing Timing Prediction (Time) [45]	Computational Graph (CG) [150]	Operational Amplifiers (AMP) [29]
Type		digital	digital	digital	digital	analog
Level		graph	node	node	graph	graph
Target		regression	classification	regression	regression	regression
Task		LUT, DSP, CP	node shared by MAJ and XOR, root node of an adder	hold slack, setup slack	CPU/GPU630/GPU640	gain, PM, BW
Evaluation Metric		mse, r2	accuracy, f1 recall, precision	mse, r2	rmse, acc5, acc10	mse, rmse
In-Distribution		CDFG	24-bit	graph structure	network structure	stage3
Out-of-Distribution		DFG	32, 36, 48- bit	graph structure	network structure	stage2
# Training Graph		16570 - 16570	1 - 1	7 - 7	5* - 10000	7223-7223
#Train Nodes	average	95	4440	29839	218	9
	max	474	4440	58676	430	16
# Train Edges	average	123	10348	41268	240	15
	max	636	10348	83225	487	36

Table 1: Statistics of selected datasets. In row '# Training graph', we report '# Graph Structures - # Samples'. *: in CG, there are only five unique CNN designs, yet the structure of graphs within each design may vary slightly.

distribution shift to simulate potential OOD challenges. We observe that while ML models perform reasonably well on tasks (8 of 13) with diverse graph structures in the training dataset, they generally suffer from OOD generalization issues on the remaining tasks. This finding highlights the urgent need for future research to focus on improving the OOD generalization capabilities of DGRL models.

Lastly, our benchmark is implemented with a modular and user-friendly codebase, allowing hardware practitioners to evaluate all 21 DGRL models for their tasks with data in a PyG-compatible format [38], and allowing ML researchers to advance DGRL methods using the collected hardware design tasks.

2 Related Work

Graph Representation Learning as Powerful Surrogate Models. ML-based surrogate models have been widely adopted in scientific fields [96, 157] and recently extended in hardware design. While graph-learning-based surrogate models for hardware design have already demonstrated effectiveness [10, 11, 14, 71, 82, 85, 102, 120, 121, 128, 136, 137, 149], several aspects warrant further investigation. First, existing studies often rely on task-specific heuristics to encode circuit structural information [10, 14, 85, 91, 102, 121], hindering the migration of model-design insights from one task to an even closely related task. Second, the majority of these studies conduct message passing of GNNs along edge directions, with few considering BI implementation [45, 51], and there is an absence of a comparative analysis of different DGRL approaches. Third, the designed models are often trained and tested within similar data distributions [10, 51, 153], lacking systematic OOD evaluation for new or more complicated designs. Hence, it is imperative to establish a comprehensive benchmark to compare different DGRL approaches for hardware design tasks.

Methods for DGRL. NN architectures for DGRL can be classified into three types: spatial GNNs, spectral GNNs, and transformers. Spatial GNNs use graph topology as inductive bias, some employ bidirected message passing for regular directed graphs [57, 65, 104, 131], others use asynchronous message passing exclusively designed for directed acyclic graphs (DAGs) [30, 116, 151]. Spectral GNNs generalize the ideas of Fourier transform and corresponding spectral convolution from undirected to directed graphs [39, 40, 49, 64, 84, 94, 110, 119, 152]; Transformers with attention mechanism reply on designing direction-aware PEs to capture directed graph topology. This benchmark is the first to consider combining transformers with MPNN layers for DGRL, extending the ideas in [101]. Regarding the choices of PEs, most studies are on undirected graphs [33, 53, 74, 127]. For

directed graphs, the potential PEs are Laplacian eigenvectors of the undirected graphs by symmetrizing the original directed ones [32], singular vectors of adjacency matrices [55] and the eigenvectors of Magnetic Laplacians [36, 37, 42, 109]. No previous investigate benefit for DGRL from stably incorporating PE [53, 127], and we are the first to consider stable PEs for DGRL.

Existing Relevant Benchmarks. Dwivedi et al. [34] benchmark long-range reasoning of GNNs on undirected graphs; PyGSD [50] benchmarks signed and directed graphs, while focusing on social or financial networks. We also compare all the methods for directed unsigned graphs in PyGSD and notice that the SOTA spectral method therein - MagNet [152] still works well on node-level tasks on a single graph (SR), which shares some similar insights. The hardware community has released graph-structured datasets from various development stages to assist surrogate model development, including but not limited to NN workload performance [100, 150], CPU throughput [20, 89, 114], resource and timing in HLS [11, 137], design quality in logic synthesis [24], design rule checking in physical synthesis [17, 21, 45, 143], and hardware security [148]. In addition to datasets, ProGraML [26] introduces a graph-based representation of programs derived from compiler IRs (e.g., LLVM/XLA IRs) for program synthesis and compiler optimization. Very recently, Google launched TPUgraph for predicting the runtime of ML models based on their computational graphs on TPUs [100]. Our CG dataset includes computational graphs of ML models, specifically on edge devices.

3 Datasets and Tasks

This section introduces the five datasets with thirteen tasks used in this benchmark. The datasets cover both digital and analog hardware, considering different circuit abstraction levels, as illustrated in Fig. 1. Table 1 displays the statistics of each dataset. Next, we briefly introduce the five datasets, with details provided in Appendix. D. Although these datasets are generated by existing studies, we offer modular pre-processing interfaces to make them compatible with PyTorch Geometric and user-friendly for integration with DGRL methods.

High-Level Synthesis (HLS) [137]: The HLS dataset collects IR graphs of C/C++ code after front-end compilation [9], and provides post-implementation performance metrics on FPGA devices as labels for each graph, which are obtained after hours of synthesis with Vitis [5] and implementation with Vivado [6]. The labels to predict include resource usage, (i.e., look-up table (LUT) and digital signal processor (DSP)), and the critical path timing (CP). See Appendix. D.1 for graph input details.

Significance: The HLS dataset is crucial for testing NNs' ability to accurately predict post-implementation metrics to accelerate design evaluation in the stage of HLS.

OOD Evaluation: For training and ID testing, we use control data flow graphs (CDFG) that integrate control conditions with data dependencies, derived from general C/C++ code; As to OOD cases, we use data flow graphs (DFG) derived from basic blocks, leading to distribution shifts.

Symbolic Reasoning (SR) [134]: The SR dataset collects bit-blasted Boolean networks (BNs) (unstructured gate-level netlists), with node labels annotating high-level abstractions on local graph structures, e.g., XOR functions, majority (MAJ) functions, and adders, generated by the logic synthesis tool ABC [13]. Each graph supports two tasks: root nodes of adders, and nodes shared by XOR and MAJ functions. See Appendix. D.2 for detailed input encoding and label explanation.

Significance: Reasoning high-level abstractions from BNs has wide applications in improving functional verification efficiency [25] and malicious logic identification [87]. GNN surrogate models are anticipated to replace the conventional structural hashing and functional propagation [70, 112] and boost the scalability with significant speedup. For graph ML, due to significant variation in the size of gate-level netlists under different bit widths, SR is an ideal real-world application to evaluate whether GNN designs can maintain performance amidst the shifts in graph scale.

OOD Evaluation: We use a 24-bit graph (4440 nodes) for training, and $32, 36, 48$-bit graphs (up to 18096 nodes) for ID testing, derived from carry-save-array multipliers before technology mapping. OOD testing data are multipliers after ASAP 7nm technology mapping [141] with the same bits.

Pre-routing Timing Prediction (TIME) [45]: The TIME dataset collects real-world circuits with OpenROAD [3] on SkyWater 130nm technology [4]. The goal is to predict slack values at timing

endpoints for each circuit design by using pre-routing information. Two tasks are considered: hold slack and setup slack. Details are provided in Appendix. D.3.

Significance: In physical synthesis, timing-driven placement demands accurate timing information, which is only available after routing. Repetitive routing and static timing analysis provide accurate timing but are prohibitively expensive. ML models that precisely learn routing behaviors and timing computation flows are highly expected to improve the efficiency of placement and routing.

OOD Evaluation: We divide ID-OOD based on the difference in graph structures (e.g. 'blabla' and 'xtea' are different circuit designs, allocated into ID or OOD groups). See details in Appendix. D.3.1.

Computational Graph (CG) [150]: The CG dataset consists of computational graphs of convolutional neural networks (CNNs) with inference latency on edge devices (i.e., Cortex A76 CPU, Adreno 630 GPU, Adreno 640 GPU) as labels. The CNNs have different operator types or configurations, either manually designed or found by neural architecture search (NAS). Details are in Appendix. D.4.

Significance: Accurately measuring the inference latency of DNNs is essential for high-performance deployment on hardware platforms or efficient NAS [103, 106], which however is often costly. ML-based predictors offer the potential for design exploration and scaling up to large-scale hardware platforms.

OOD Evaluation: We split ID-OOD with different graph structures. (e.g. 'DenseNets' and 'ResNets' are CNNs with different structures, allocated into different groups). See Appendix. D.4.1 for details.

Operational Amplifiers (AMP) [29]: AMP dataset contains $10,000$ distinct 2- or 3-stage operational amplifiers (Op-Amps). Circuit specifications (i.e. DC gain, phase margin (PM), and bandwidth (BW)) as labels are extracted after simulation with Cadence Spectre [1]. Details are in Appendix. D.6.

Significance: Analog circuit design is less automated and requires more manual effort compared to its digital counterpart. Mainstream approaches such as SPICE-based circuit synthesis and simulation [124], are computationally expensive and time-consuming. If ML algorithms can approximate the functional behavior and provide accurate estimates of circuit specifications, they may significantly reduce design time by minimizing reliance on circuit simulation [7].

OOD Evaluation: For training and ID testing, we use 3-stage Op-Amps, which have three single-stage Op-Amps in the main feed-forward path). For OOD evaluation, we use 2-stage Op-Amps.

Extensions Although the datasets cover different levels of circuit abstraction, there are additional tasks in hardware design worth exploration with DGRL surrogates, as reviewed in Section 2. Our modular benchmark framework allows for easy extension to accommodate new datasets.

4 Benchmark Design

4.1 Design Space for Directed Graph Representation Learning

In this section, we introduce the DGRL methods evaluated in this benchmark. Our evaluation focuses on four design modules involving GNN backbones, message passing directions, transformer selection, and PE incorporation, illustrated in Fig. 2. Different GNN backbones and transformer adoptions cover 10 methods in total with references in Tab. 2. We also consider their combinations with different message-passing directions and various ways to use PEs, which overall gives 21 DGRL methods.

For GNNs, we consider 4 spectral methods, namely GCN [63], DGCN [119], DiGCN [118] and MagNet [152], where the latter three are *SOTA spectral GNNs* specifically designed for DGRL [50]; For spatial GNNs, we take GIN [140] and Graph Attention Network (GAT) [123], which are the most commonly used MPNN backbones for undirected graphs. We evaluate the combination of GCN, GIN and GAT with three different message-passing directions: a) 'undirected'(-) treats directed graphs as undirected, using the same NN parameters to perform message-passing along both forward and reverse edge directions; b) 'directed'(DI) only passes messages exclusively along the forward edge directions; c) 'bidirected'(BI) performs message passing in both forward and reverse directions with distinct parameters for either direction. The other GNNs (DGCN, DiGCN and MagNet) adopt

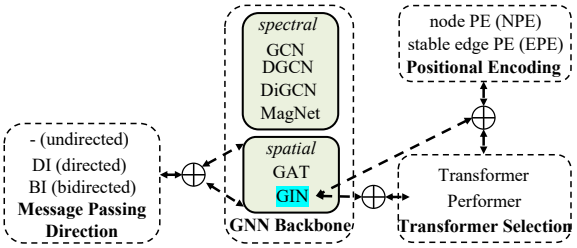

Figure 2: The benchmark considers 21 combinations of message passing direction, GNN backbone, transformer selection and PE incorporation, covers 10 existing SOTA methods from graph ML community and discovers 2 novel top-performing models (see Table. 2).

Method	type	layer-wise complexity
GCN [63]	spectral	$O(\|E\|)$
MagNet [152]	spectral	$O(\|E\|)$
DGCN [119]	spectral	$O(\|E\|)$
DiGCN [118]	spectral	$O(\|E\|)$
GAT [123]	spatial	$O(\|E\|)$
GIN(E) [140]	spatial	$O(\|E\|)$
EDGNN [57]	spatial	$O(\|E\|)$
GPS-T [101]	spatial+transformer	$O(\|V\|^2+\|E\|)$
GPS-P [23]	spatial+transformer	$O(\|V\|+\|E\|)$
TmD [42]	transformer	$O(\|V\|^2)$
BI-GIN(E)+EPE(new)	spatial	$O(\|E\|)$
BI-GPS-T+EPE(new)	spatial+transformer	$O(\|V\|^2+\|E\|)$

Table 2: Existing methods and two top-performing methods highlighted at bottom.

spectral convolution that inherently considers edge directions. The combination of 'BI' with spatial GNN layers gives *the state-of-the-art spatial GNNs* for DGRL, i.e., EDGNN [57].

For GTs, we adopt the eigenvectors of the graph Magnetic Laplacian (MagLAP) matrix as the PEs of nodes [40, 109], as they are directional-aware. The MagLap matrix $\boldsymbol{L}_q$ is a complex Hermitian matrix with parameter $q \in [0, 1)$ named potential, which is treated as a hyper-parameter in our experiments. Note that when $q = 0$, MagLap degenerates to the symmetric Laplacian matrix $\boldsymbol{L}_0$ as a special case. See Appendix B for a brief review of MagLap. The GT with the MagLap PEs attached to node features gives *the SOTA GT model* for DGRL, named TmD for brevity, proposed in [42]. GPS [101] is a GT model with MPNN layers [43, 48] interleaving with transformer layers [122], originally proposed for undirected graphs. We extend GPS to directed graphs by using MagLap PEs for transformer layers and DI/BI message passing in its MPNN layers. Hence, GPS is also an extension of TmD by incorporating MPNN layers. As transformers may not scale well on large graphs, we evaluate vanilla transformer layers and their lower-rank approximation Performer [67] for efficient computation, named as GPS-T and GPS-P, respectively.

4.2 Stable Direction-aware Positional Encodings

Recent studies on undirected graphs have demonstrated that models by naively attaching PEs to node features may suffer from an issue of instability because small changes in the graph structure may cause big changes in PEs [53, 74, 127]. We name this way of using PEs as node-PE (NPE). The instability provably leads to undesired OOD generalization [53]. We think this is also true for directed graphs and indeed observe the subpar model performance with NPE.

$\textbf{NPE} = [\text{Re}\{\boldsymbol{V}_q\}, \text{Im}\{\boldsymbol{V}_q\}]$
$\textbf{EPE} = \rho(\text{Re}\{\boldsymbol{V_q}\text{diag}(\kappa_1(\lambda))\boldsymbol{V_q}^\dagger\}, ..., \text{Re}\{\boldsymbol{V_q}\text{diag}(\kappa_m(\lambda))\boldsymbol{V_q}^\dagger\},$ $\text{Im}\{\boldsymbol{V_q}\text{diag}(\kappa_1(\lambda))\boldsymbol{V_q}^\dagger\}, ..., \text{Im}\{\boldsymbol{V_q}\text{diag}(\kappa_m(\lambda))\boldsymbol{V_q}^\dagger\})$

Table 3: Functions to obtain PEs. NPE directly concatenates the eigenvectors to node features. In contrast, before concatenating PE to the edge features, EPE employs the permutation equivariant functions $\kappa : \mathbb{R}^d \to \mathbb{R}^d$ w.r.t. eigenvalue permutations and permutation equivariant function $\rho : \mathbb{R}^{|V|\times|V|\times 2m} \to \mathbb{R}^{|V|\times|V|\times d}$ to stably process the eigenvectors and eigenvalues, respectively.

Therefore, besides NPE, we also consider a stable way of incorporating PEs for DGRL, namely 'edge PE' (EPE), inspired by [127]. EPE was originally proposed for the undirected graph case. Specifically, we use the smallest d eigenvalues $\lambda_q \in \mathbb{R}^d$ and their corresponding eigenvectors $\boldsymbol{V}_q \in \mathbb{C}^{|V|\times d}$ from $\boldsymbol{L}_q$. Then, we follow the equation in Table 3 to compute $\textbf{EPE} \in \mathbb{R}^{|V|\times|V|\times d}$. Then, in GTs, $\textbf{EPE}_{u,v}$ is further added to the attention weight between nodes u and v as a bias term at each attention layer.

We note that PEs can also be used in more than GTs, to improve the expressive power of GNNs [53, 69, 74, 145]. We leverage this idea and enhance the GNN models for directed graphs with PEs. Specifically, for the GNNs NPE will use $\textbf{NPE}_v$ as extra node features of node v while EPE will use $\textbf{EPE}_{u,v}$ as extra edge features of edge uv if uv is an edge.

The incorporation with EPE helps discover a novel GT model for directed graphs, i.e., GT with BI-MPNN layers enhanced by EPE, abbreviated as BI-GPS+EPE. We also make the first attempt to combine GNNs with PEs for directed graphs, which yields the model BI-GIN(E)+EPE.

4.3 Hyer-Parameter Space and Tuning

For each combination of DGRL method in this benchmark, we perform automatic hyper-parameter tuning with RAY [73] adopting Tree-structured Parzen Estimator (TPE) [130], a state-or-the-art bayesian optimization algorithm. The hyper-parameter space involves searching batch size, learning rate, number of backbone layers, dropout rate in MPNN and MLP layers, hidden dimension, and MLP layer configurations. The detailed hyper-parameter space of each model is shown in Appendix. E.2. We auto-tune the hyper-parameters with seed 123 with 100 trial budgets and select the configuration with the best validation performance. Then, the selected configuration is used for model training and testing ten times with seeds $0-9$ and the average is reported as the final performance.

5 Modular Toolbox

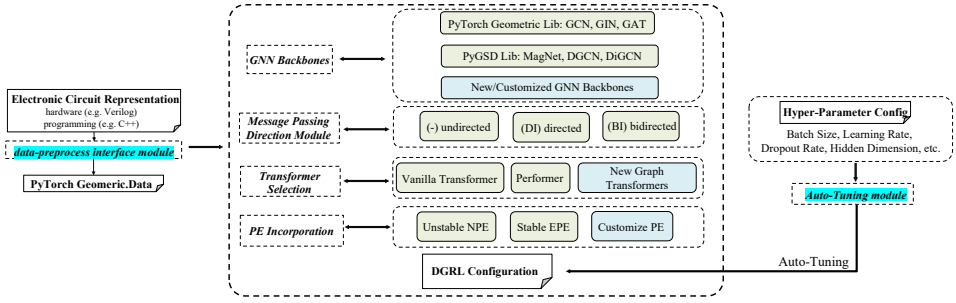

Figure 3: Illustration of the directed graph representation learning (DGRL) toolbox.

We develop a highly modular toolbox involving designing, auto hyper-parameter tuning, and evaluation for DGRL methods. The framework is shown in Fig. 3. The toolbox comes with the 21 DGRL methods, allowing practitioners to evaluate them on any new task with data compatible with PyTorch Geometric (PyG) [38]. This may be used even beyond hardware design applications. Users can also customize new methods. Once the method is configured, auto hyper-parameter tuning can be performed using RAY [73]. The toolbox also includes the above 5 datasets with 13 tasks that can be used to develop new DGRL models. For details please refer to the official document for this toolbox.

6 Experiments

In this section, we first evaluate DGRL methods combining different GNN backbones, message passing directions, transformer selection, and PE incorporation, across all 5 datasets and 13 tasks, using in-distribution (ID) and out-of-distribution (OOD) testing data.

6.1 Main Results

The performances of the methods under all evaluation metrics for both in-distribution and out-of-distribution testing across all 13 tasks are reported from Table. 11 to Table. 33 in Appendix. G.1. We summarize the averaged ranking with respect to all evaluation metrics given a task in Table. 4. The details of ranking calculation is in Appendix. F.1. The results tell the following insights:

'Bidirected' (BI) message passing in the MPNN layers significantly boosts the models' performance on three GNN backbones (GCN, GIN, GAT) and one GT backbone (GPS-T): BI-GCN outperforms GCN on 10 out of 13 tasks in both ID and OOD evaluations. Similarly, in ID/OOD evaluations, BI-GIN outperforms GIN in 11/12 out of 13 tasks, BI-GAT outperforms GAT in 11/9 out of 13 tasks and BI-GPS-T outperforms GPS-T in 5/5 out of 6 tasks, respectively.

Distribution		In-Distribution (ID)												Out-of-Distribution (OOD)											
Dataset		HLS			AMP			SR		TIME		CG			HLS			AMP			SR		TIME		CG	
Task		DSP	LUT	CP	gain	PM	BW	share	root	hold	setup	CPU	GPU630	GPU640	DSP	LUT	CP	gain	PM	BW	share	root	hold	setup	CPU	GPU630	GPU640
Spectral	DGCN	15.0	15.0	15.0	14.0	8.0	15.0	10.0	9.0	15.0	5.5	13.0	15.0	14.0	15.0	14.0	15.0	14.0	3.0	15.0	7.5	5.0	15.0	7.0	13.3	11.7	11.2
	DiGCN	12.0	14.0	13.0	12.0	9.0	14.0	8.5	7.8	13.5	15.0	14.0	14.0	15.0	12.5	15.0	14.0	9.0	4.0	14.0	9.0	5.0	13.5	14.0	13.2	13.2	13.3
	MagNet	7.0	7.0	10.5	8.0	11.0	8.0	1.8	2.0	11.0	11.5	1.3	1.3	4.7	7.0	7.0	10.5	3.0	12.0	8.0	3.5	8.8	9.0	7.0	4.2	8.2	7.3
	GCN	14.0	12.0	14.0	15.0	13.0	12.0	13.3	13.5	9.5	14.0	15.0	12.3	11.7	12.5	10.0	12.0	14.5	14.0	11.0	14.8	14.5	7.5	10.5	12.7	12.7	11.5
	DI-GCN	13.5	13.0	12.0	11.0	3.0	13.0	15.0	15.0	11.0	13.0	11.0	11.3	12.0	14.0	11.0	13.0	12.0	7.0	13.0	13.5	11.8	10.0	8.0	11.2	11.5	12.2
	BI-GCN	11.0	10.5	9.0	5.0	14.0	6.0	5.5	5.3	5.0	9.0	12.3	12.3	12.3	11.0	12.5	8.0	2.0	13.0	5.0	2.3	4.8	3.0	6.5	13.2	11.3	12.5
Spatial	GIN	6.0	5.5	8.0	7.0	6.0	10.0	10.0	11.0	1.0	3.0	5.0	3.3	8.3	6.0	3.5	5.0	8.0	10.0	7.0	9.0	7.3	3.0	8.5	5.2	4.2	4.8
	DI-GIN	2.5	4.0	6.5	9.0	10.0	7.0	6.5	4.8	3.0	2.0	5.7	8.0	3.3	2.0	2.5	7.0	10.0	5.0	12.0	6.3	9.0	6.5	7.0	3.5	5.7	4.2
	BI-GIN	1.0	1.0	5.0	3.0	4.0	3.0	2.8	4.8	2.0	1.0	2.3	4.7	1.0	4.5	1.0	3.0	4.0	9.0	3.0	1.5	2.0	1.0	7.5	2.3	4.5	4.0
	GAT	8.5	9.0	6.5	6.0	15.0	5.0	13.8	13.5	10.5	11.5	9.0	9.0	8.7	9.0	9.0	5.5	7.0	15.0	6.0	12.3	10.5	10.5	5.5	7.7	5.7	6.2
	DI-GAT	10.0	10.5	10.5	10.0	12.0	9.0	11.8	10.0	13.5	10.0	10.0	10.0	10.0	10.0	12.5	11.0	11.0	6.0	10.0	11.3	10.0	13.5	8.5	6.2	5.7	7.3
	BI-GAT	9.0	8.0	1.0	4.0	2.0	11.0	4.0	6.3	6.5	7.5	8.0	5.3	7.0	8.0	8.0	1.5	1.0	2.0	9.0	9.8	8.5	4.5	6.5	6.2	10.7	10.5
Transformer	GPS-T	4.0	3.0	2.5	13.0	7.0	2.0	--	--	--	--	--	--	--	5.0	6.0	9.5	13.0	8.0	1.0	--	--	--	--	--	--	--
	DI-GPS-T	5.0	5.5	4.0	1.0	5.0	1.0	--	--	--	--	--	--	--	3.0	5.0	1.5	5.0	11.0	2.0	--	--	--	--	--	--	--
	BI-GPS-T	3.0	2.0	2.0	1.0	1.0	4.0	--	--	--	--	--	--	--	2.5	3.5	4.5	6.0	1.0	4.0	--	--	--	--	--	--	--
	GPS-P	--	--	--	--	--	--	5.5	12.0	6.5	4.0	6.3	2.0	5.3	--	--	--	--	--	--	7.8	11.3	8.0	7.5	6.2	4.2	6.2
	DI-GPS-P	--	--	--	--	--	--	6.5	4.8	4.0	7.5	2.7	5.7	3.0	--	--	--	--	--	--	5.8	7.5	7.5	8.0	7.5	7.0	5.8
	BI-GPS-P	--	--	--	--	--	--	7.8	7.5	8.0	5.5	4.7	6.0	4.3	--	--	--	--	--	--	6.8	4.3	7.5	8.0	8.5	5.2	4.2

Table 4: Average ranking (↓) of methods across datasets/tasks/metrics on ID and OOD data.

Distribution	In-Distribution (ID)												Out-of-Distribution (OOD)											
Dataset	HLS			AMP			SR		TIME		CG			HLS			AMP			SR		TIME		CG	
Task	DSP	LUT	CP	gain	PM	BW	share	root	hold	setup	CPU	GPU630	GPU640	DSP	LUT	CP	gain	PM	BW	share	root	hold	setup	CPU	GPU630	GPU640
MagNet	14.5	11.0	14.5	12.0	15.0	12.0	2.3	2.5	13.0	13.0	2.3	1.7	6.7	11.0	11.0	14.5	3.0	16.0	12.0	5.5	10.8	11.0	16.0	5.2	9.5	8.3
BI-GIN(E)	9.0	2.0	9.0	6.0	6.0	6.0	4.8	6.8	2.5	2.0	6.0	3.3	6.0	7.5	3.5	6.0	4.0	13.0	5.0	3.0	3.0	2.5	7.5	4.7	5.8	4.8
BI-GIN(E)+NPE	5.0	4.0	5.0	5.0	13.0	5.0	3.0	6.8	5.5	5.0	8.3	5.0	5.3	9.0	4.0	1.5	7.0	8.0	7.0	2.8	3.5	5.5	12.5	6.0	4.5	6.0
BI-GIN(E)+EPE	5.0	1.0	5.0	9.0	10.0	3.0	4.0	6.8	2.0	1.0	1.0	6.7	1.7	7.0	1.0	2.5	6.0	6.0	4.0	1.5	3.0	1.0	7.5	3.8	5.5	4.7
BI-GPS-T (NPE)	4.5	5.5	4.5	2.0	2.0	7.0	--	--	--	--	--	--	--	4.0	7.0	8.0	9.0	2.0	6.5	--	--	--	--	--	--	--	--
BI-GPS-T+EPE	2.5	3.0	2.0	1.0	1.0	4.0	--	--	--	--	--	--	--	1.5	1.5	3.5	5.5	1.0	1.0	--	--	--	--	--	--	--	--

Table 5: Comparison of competitive methods involving NPE and EPE. The ranking (↓) is based on all the 18 methods in Table 4 plus BI-GIN(E)+NPE, BI-GIN(E)+EPE and BI-GPS-T+EPE.

As to the models, on datasets with small graphs (HLS and AMP), BI-GPS-T consistently delivers excellent results, achieving top-3 performance in 5 out of 6 tasks on both ID and OOD testing data. BI-GIN also demonstrates competitive performance on these datasets. However, for datasets with larger graphs (SR, CG, and TIME), BI-GPS-T encounters a scalability issue. BI-GIN secures top-three performance in 6 out of 7 tasks in both ID and OOD testing data. For the 'shared' and 'root' tasks from the SR dataset and the 'CPU' and 'GPU630' tasks from the CG dataset, MagNet [152] performs best in the ID setting. This is likely because training and testing are conducted on the same graph structures for these specific datasets, reducing the need for significant generalization across different graph structures. This scenario aligns well with the spectral filtering approach used by MagNet. These observations match findings from previous studies on directed networks [50, 152]. However, MagNet's performance falters in OOD evaluations which ask for the ability to generalize across different graph structures. GPS-P, despite its capability to handle large graphs, delivers only mediocre performance overall. *In conclusion, BI-GPS is well-suited for small (around one hundred nodes) directed graphs. For larger graphs, BI-GIN is efficient and performs well. For tasks where the training and testing data share the same graph structures, one may also attempt to adopt MagNet.*

Comparing PE-enhanced methods: We further investigate the impact of different ways of using PEs. We combine NPE or EPE with the top-performing models from the previous section and evaluate BI-GIN+NPE, BI-GIN+EPE, and BI-GPS+EPE. Note that BI-GPS already utilizes NPE. We have chosen not to consider adding PE to MagNet because MagNet only accepts 1-dimensional edge weights, limiting its ability to leverage EPE. We provide a summary of the performance data from Table 34 to Table 43 in Appendix G.2 and report the average rankings of the methods for each task. All 18 methods in Table 4, along with the 3 new combinations, are included in the ranking. We detail the results of the most competitive methods in Table 5. For BI-GIN, EPE enhances its performance on 10 out of 13 tasks in the in-distribution (ID) testing data and 11 tasks in the out-of-distribution (OOD) testing data. Conversely, NPE only improves the performance of BI-GIN on 7 tasks in the ID testing and 4 tasks in the OOD testing and performs unstable for the rest tasks. Notably, EPE-enhanced BI-GIN surpasses MagNet on the CPU task in the CG dataset. For BI-GPS-T, EPE improves its performance on all 6 tasks in both ID and OOD testing, while NPE does not yield substantial improvements. This observation contrasts with previous work [101] on undirected graphs for molecular property prediction. *In conclusion, we find that incorporating PEs in a stable way as EPE significantly boosts the performance of different models across the selected tasks and datasets.*

dataset (baseline's name)	AMP [29] (CKTGNN)			HLS [137] (Hierarchical GNN)			SR [134] (GAMORA)	CG [150] (nn-meter)			TIME [45] (Timer-GNN)
task	gain	PM	BW	dsp	lut	cp	shared	cpu (average)			hold
metric	rmse↓	rmse↓	rmse↓	mse↓	mse↓	mse↓	accuracy↑	rmse↓	acc5↑	acc10↑	r2↑
Baseline	0.52	1.15	4.47	3.94	2.45	0.88	0.99	3.20	0.80	0.99	0.97
BI-GINE+EPE	0.51±0.07	1.14±0.00	4.20±0.13	2.13±0.08	1.73±0.10	0.61±0.02	0.99±0.00	2.79±0.14	0.86±0.02	0.99±0.01	0.99±0.00
BI-GPS-T+EPE	0.34±0.08	1.15±0.00	3.79±0.11	2.13±0.15	1.96±0.13	0.60±0.01	- -	- -	- -	- -	- -

Table 6: Comparison of BI-GIN+EPE and BI-GPS-T+EPE with baselines specific for each dataset.

6.2 Summary: The Recipe for DGRL

Through benchmarking various combinations within the design space, we have formulated a design recipe for DGRL methods tailored for encoding hardware data: *The use of 'bidirected' (BI) message passing and stable positional encodings (PE) can significantly enhance model performance. Therefore, we recommend BI-GPS-T+EPE for encoding small graphs and BI-GIN+EPE for large graphs.*

We further compare the two models' performance with the baseline methods proposed by hardware design practitioners specifically for the corresponding tasks in the original papers. Results are shown in Table. 6. The comparison focuses on ID evaluation as for most of the tasks, the original studies did not even report OOD evaluations. We follow the same data split as baseline methods for fair comparison (see the details in Appendix C). BI-GIN+EPE achieves results comparable to, or better than, the baseline methods. BI-GPS+EPE achieves even better performance than BI-GIN+EPE for small graphs. Note that the baseline methods for certain tasks may incorporate domain-specific expert knowledge and additional data processing. For example, CKTGNN [29] for the AMP dataset modifies the graph structures into DAGs and employs an asynchronized message passing to mimic the signal flow in these amplifiers; 'timer-GNN' [45] is tailored for the TIME dataset to mimic the transmission rules of clock signals and designs a non-linear delay model (NLDM) along with a novel module 'cell library'. Such domain knowledge may further enhance BI-GPS+EPE and BI-GIN+EPE for these specific tasks, which is left for future research.

Discussion on OOD Evaluation: Despite BI-GPS-T+EPE and BI-GIN+EPE outperforming other methods in OOD testing across all tasks, we cannot yet conclude that these methods are sufficiently effective for practical OOD usage. *In fact, making accurate predictions with OOD data in hardware design remains a significant challenge.* When the graph structures in training sets are sufficiently diverse, such as in datasets with a large number of small graphs (e.g., AMP, HLS) or those with abundant local structures (e.g., SR), BI-GIN+EPE and BI-GPS-T+EPE tend to maintain reasonably good performance on OOD data. However, OOD generalization becomes challenging when the diversity of graph structures in the training set is limited. For instance, in the TIME dataset, which has a limited variety of graph structures for training and OOD testing data with entirely different graph structures, both BI-GIN+EPE and BI-GPS-T+EPE perform worse than timer-GNN [45], which integrates the knowledge of the physical structure of circuits (as shown in Table 21). We identify ensuring OOD performance, especially when training sets lack sufficiently diversified graph structures, as a key direction for future DGRL research.

7 Conclusions and Limitations

Through benchmarking 21 methods on in-distribution and out-of-distribution test sets across 13 tasks and 5 datasets within the hardware design loop, we find bidirected (BI) message passing neural networks can substantially improve the performance of both Graph Transformer (GT) encoders that incorporate MPNN layers and pure GNN encoders. Positional Encodings (PEs), particularly when used stably, can broadly enhance the performance of both GTs and GNNs. With these insights, we identify two top-performing models: BI-GPS-T+EPE and BI-GIN+EPE, both of which outperform the baseline models originally proposed for the corresponding tasks.

Limitations: Although the benchmark covers multiple stages in hardware design loop, there are other tasks [10, 17, 20, 89, 114, 143, 149] that could be included in this benchmark as DGRL tasks. Given technological advancements and the diversity of design tools, ensuring OOD performance remains an urgent open problem in hardware design. Future research may involve high-quality data collection [46, 56, 132, 138, 139] or the development of OOD-aware DGRL methods [78–80, 105].

References

[1] Cadence spectre simulation platform. https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation.html.

[2] Opencores. https://opencores.org/.

[3] Openroad. https://github.com/The-OpenROAD-Project/OpenROAD.

[4] Skywater. https://github.com/google/skywater-pdk.

[5] Vitis hls tool. https://www.xilinx.com/products/design-tools/vitis/vitis-hls.html.

[6] Vivado. https://www.xilinx.com/products/design-tools/vivado.html.

[7] Engin Afacan, Nuno Lourenço, Ricardo Martins, and Günhan Dündar. Machine learning techniques in analog/rf integrated circuit design, synthesis, layout, and test. *Integration*, 77:113–130, 2021.

[8] Abeer Al-Hyari, Hannah Szentimrey, Ahmed Shamli, Timothy Martin, Gary Grewal, and Shawki Areibi. A deep learning framework to predict routability for fpga circuit placement. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 14(3):1–28, 2021.

[9] V Aho Alfred, S Lam Monica, and D Ullman Jeffrey. *Compilers Principles, Techniques & Tools*. pearson Education, 2007.

[10] Lilas Alrahis, Abhrajit Sengupta, Johann Knechtel, Satwik Patnaik, Hani Saleh, Baker Mohammad, Mahmoud Al-Qutayri, and Ozgur Sinanoglu. GNN-RE: Graph neural networks for reverse engineering of gate-level netlists. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pages 1–1, 2021.

[11] Yunsheng Bai, Atefeh Sohrabizadeh, Zongyue Qin, Ziniu Hu, Yizhou Sun, and Jason Cong. Towards a comprehensive benchmark for high-level synthesis targeted to fpgas. *Advances in Neural Information Processing Systems*, 36:45288–45299, 2023.

[12] Ioana Baldini, Stephen J Fink, and Erik Altman. Predicting gpu performance from cpu runs using machine learning. In *2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing*, pages 254–261. IEEE, 2014.

[13] Robert Brayton and Alan Mishchenko. Abc: An academic industrial-strength verification tool. In *Proc. CAV*. Springer, 2010.

[14] Tim Bücher, Lilas Alrahis, Guilherme Paim, Sergio Bampi, Ozgur Sinanoglu, and Hussam Amrouch. Appgnn: Approximation-aware functional reverse engineering using graph neural networks. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[15] Burcin Cakir and Sharad Malik. Reverse engineering digital ics through geometric embedding of circuit graphs. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 23(4):1–19, 2018.

[16] Weidong Cao, Mouhacine Benosman, Xuan Zhang, and Rui Ma. Domain knowledge-based automated analog circuit design with deep reinforcement learning. *arXiv preprint arXiv:2202.13185*, 2022.

[17] Zhuomin Chai, Yuxiang Zhao, Wei Liu, Yibo Lin, Runsheng Wang, and Ru Huang. Circuitnet: An open-source dataset for machine learning in vlsi cad applications with improved domain-specific evaluation metric and learning strategies. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[18] Jingsong Chen, Jian Kuang, Guowei Zhao, Dennis J-H Huang, and Evangeline FY Young. Pros: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2020.

[19] Tianqi Chen, Lianmin Zheng, Eddie Yan, Ziheng Jiang, Thierry Moreau, Luis Ceze, Carlos Guestrin, and Arvind Krishnamurthy. Learning to optimize tensor programs. *Advances in Neural Information Processing Systems*, 31, 2018.

[20] Yishen Chen, Ajay Brahmakshatriya, Charith Mendis, Alex Renda, Eric Atkinson, Ondrej Sykora, Saman Amarasinghe, and Michael Carbin. Bhive: A benchmark suite and measurement framework for validating x86-64 basic block performance models. In *2019 IEEE international symposium on workload characterization (IISWC)*. IEEE, 2019.

[21] Vidya A Chhabria, Wenjing Jiang, Andrew B Kahng, Rongjian Liang, Haoxing Ren, Sachin S Sapatnekar, and Bing-Yue Wu. Openroad and circuitops: Infrastructure for ml eda research and education. In *2024 IEEE 42nd VLSI Test Symposium (VTS)*, pages 1–4. IEEE, 2024.

[22] Matteo Chinazzi and Giorgio Fagiolo. *Systemic risk, contagion, and financial networks: A survey*. SSRN, 2015.

[23] Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. 2020.

[24] Animesh Basak Chowdhury, Benjamin Tan, Ramesh Karri, and Siddharth Garg. Openabc-d: A large-scale dataset for machine learning guided integrated circuit synthesis. *arXiv preprint arXiv:2110.11292*, 2021.

[25] Maciej Ciesielski, Tiankai Su, Atif Yasin, and Cunxi Yu. Understanding algebraic rewriting for arithmetic circuit verification: a bit-flow model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(6):1346–1357, 2019.

[26] Chris Cummins, Zacharias Fisches, Tal Ben-Nun, Torsten Hoefler, Michael O'Boyle, and Hugh Leather. ProGraML: A Graph-based Program Representation for Data Flow Analysis and Compiler Optimizations. In *Thirty-eighth International Conference on Machine Learning (ICML)*, 2021.

[27] Steve Dai, Yuan Zhou, Hang Zhang, Ecenur Ustun, Evangeline FY Young, and Zhiru Zhang. Fast and accurate estimation of quality of results in high-level synthesis with machine learning. In *2018 IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 129–132. IEEE, 2018.

[28] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations*, 2019.

[29] Zehao Dong, Weidong Cao, Muhan Zhang, Dacheng Tao, Yixin Chen, and Xuan Zhang. Ckt-gnn: Circuit graph neural network for electronic design automation. *International Conference on Learning Representations*, 2023.

[30] Zehao Dong, Muhan Zhang, Fuhai Li, and Yixin Chen. Pace: A parallelizable computation encoder for directed acyclic graphs. In *International Conference on Machine Learning*, pages 5360–5377. PMLR, 2022.

[31] Lukasz Dudziak, Thomas Chau, Mohamed Abdelfattah, Royson Lee, Hyeji Kim, and Nicholas Lane. Brp-nas: Prediction-based nas using gcns. *Advances in Neural Information Processing Systems*, 33:10480–10490, 2020.

[32] Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023.

[33] Vijay Prakash Dwivedi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Graph neural networks with learnable structural and positional representations. In *International Conference on Learning Representations*, 2022.

[34] Vijay Prakash Dwivedi, Ladislav Rampášek, Mikhail Galkin, Ali Parviz, Guy Wolf, Anh Tuan Luu, and Dominique Beaini. Long range graph benchmark. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[35] Hadi Esmaeilzadeh, Soroush Ghodrati, Andrew Kahng, Joon Kyung Kim, Sean Kinzer, Sayak Kundu, Rohan Mahapatra, Susmita Dey Manasi, Sachin Sapatnekar, Zhiang Wang, et al. An open-source ml-based full-stack optimization framework for machine learning accelerators. *ACM Transactions on Design Automation of Electronic Systems*, 2023.

[36] Michaël Fanuel, Carlos M Alaíz, Ángela Fernández, and Johan AK Suykens. Magnetic eigenmaps for the visualization of directed networks. *Applied and Computational Harmonic Analysis*, 44(1):189–199, 2018.

[37] Michaël Fanuel, Carlos M Alaiz, and Johan AK Suykens. Magnetic eigenmaps for community detection in directed networks. *Physical Review E*, 95(2):022302, 2017.

[38] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[39] Stefano Fiorini, Stefano Coniglio, Michele Ciavotta, and Enza Messina. Sigmanet: One laplacian to rule them all. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 7568–7576, 2023.

[40] Satoshi Furutani, Toshiki Shibahara, Mitsuaki Akiyama, Kunio Hato, and Masaki Aida. Graph signal processing for directed graphs based on the hermitian laplacian. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part I*, pages 447–463. Springer, 2020.

[41] Douglas M Gale and Shachar Kariv. Financial networks. *American Economic Review*, 97(2):99–103, 2007.

[42] Simon Geisler, Yujia Li, Daniel J Mankowitz, Ali Taylan Cemgil, Stephan Günnemann, and Cosmin Paduraru. Transformers meet directed graphs. In *International Conference on Machine Learning*, pages 11144–11172. PMLR, 2023.

[43] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. PMLR, 2017.

[44] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, LIU Shujie, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, et al. Graphcodebert: Pre-training code representations with data flow. *International Conference on Learning Representations*, 2020.

[45] Zizheng Guo, Mingjie Liu, Jiaqi Gu, Shuhan Zhang, David Z Pan, and Yibo Lin. A timing engine inspired graph neural network model for pre-routing slack prediction. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 1207–1212, 2022.

[46] Nitin Gupta, Shashank Mujumdar, Hima Patel, Satoshi Masuda, Naveen Panwar, Sambaran Bandyopadhyay, Sameep Mehta, Shanmukha Guttula, Shazia Afzal, Ruhi Sharma Mittal, et al. Data quality for machine learning tasks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, pages 4040–4041, 2021.

[47] Gary D Hachtel and Fabio Somenzi. *Logic synthesis and verification algorithms*. Springer Science & Business Media, 2005.

[48] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[49] Yixuan He, Michael Perlmutter, Gesine Reinert, and Mihai Cucuringu. Msgnn: A spectral graph neural network based on a novel magnetic signed laplacian. In *Learning on Graphs Conference*, pages 40–1. PMLR, 2022.

[50] Yixuan He, Xitong Zhang, Junjie Huang, Benedek Rozemberczki, Mihai Cucuringu, and Gesine Reinert. Pytorch geometric signed directed: A software package on graph neural networks for signed and directed graphs. In *Learning on Graphs Conference*, pages 12–1. PMLR, 2024.

[51] Zhuolun He, Ziyi Wang, Chen Bai, Haoyu Yang, and Bei Yu. Graph learning-based arithmetic block identification. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2021.

[52] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

[53] Yinan Huang, William Lu, Joshua Robinson, Yu Yang, Muhan Zhang, Stefanie Jegelka, and Pan Li. On the stability of expressive positional encodings for graph neural networks. *International Conference on Learning Representations*, 2024.

[54] William Hughes, Sandeep Srinivasan, Rohit Suvarna, and Maithilee Kulkarni. Optimizing design verification using machine learning: Doing better than random. *arXiv preprint arXiv:1909.13168*, 2019.

[55] Md Shamim Hussain, Mohammed J Zaki, and Dharmashankar Subramanian. Global self-attention as a replacement for graph convolution. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 655–665, 2022.

[56] Abhinav Jain, Hima Patel, Lokesh Nagalapatti, Nitin Gupta, Sameep Mehta, Shanmukha Guttula, Shashank Mujumdar, Shazia Afzal, Ruhi Sharma Mittal, and Vitobha Munigala. Overview and importance of data quality for machine learning tasks. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3561–3562, 2020.

[57] Guillaume Jaume, An-phi Nguyen, María Rodríguez Martínez, Jean-Philippe Thiran, and Maria Gabrani. edgnn: a simple and powerful gnn for directed labeled graphs. *arXiv preprint arXiv:1904.08745*, 2019.

[58] Wenhao Jia, Kelly A Shaw, and Margaret Martonosi. Stargazer: Automated regression-based gpu design space exploration. In *2012 IEEE International Symposium on Performance Analysis of Systems & Software*, pages 2–13. IEEE, 2012.

[59] Zhihao Jia, Sina Lin, Mingyu Gao, Matei Zaharia, and Alex Aiken. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems*, 2:187–198, 2020.

[60] Sam Kaufman, Phitchaya Phothilimthana, Yanqi Zhou, Charith Mendis, Sudip Roy, Amit Sabne, and Mike Burrows. A learned performance model for tensor processing units. *Proceedings of Machine Learning and Systems*, 3:387–400, 2021.

[61] Arshinder Kaur, Arun Kanda, and SG Deshmukh. A graph theoretic approach for supply chain coordination. *international journal of logistics Systems and Management*, 2(4):321–341, 2006.

[62] Alexy Khrabrov and George Cybenko. Discovering influence in communication networks using dynamic graph analysis. In *2010 IEEE Second International Conference on Social Computing*, pages 288–294. IEEE, 2010.

[63] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[64] Christian Koke and Daniel Cremers. Holonets: Spectral convolutions do extend to directed graphs. In *The Twelfth International Conference on Learning Representations*, 2023.

[65] Georgios Kollias, Vasileios Kalantzis, Tsuyoshi Idé, Aurélie Lozano, and Naoki Abe. Directed graph auto-encoders. In *Proceedings of the AAAI conference on artificial intelligence*, volume 36, pages 7211–7219, 2022.

[66] Gueorgi Kossinets, Jon Kleinberg, and Duncan Watts. The structure of information pathways in a social communication network. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 435–443, 2008.

[67] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. *Advances in Neural Information Processing Systems*, 34:21618–21629, 2021.

[68] Menghao Li, Minjia Zhang, Chi Wang, and Mingqin Li. Adatune: Adaptive tensor program compilation made efficient. *Advances in Neural Information Processing Systems*, 33:14807–14819, 2020.

[69] Pan Li, Yanbang Wang, Hongwei Wang, and Jure Leskovec. Distance encoding: Design provably more powerful neural networks for graph representation learning. *Advances in Neural Information Processing Systems*, 33:4465–4478, 2020.

[70] Wenchao Li, Adria Gascon, Pramod Subramanyan, Wei Yang Tan, Ashish Tiwari, Sharad Malik, Natarajan Shankar, and Sanjit A Seshia. Wordrev: Finding word-level structures in a sea of bit-level gates. In *2013 IEEE international symposium on hardware-oriented security and trust (HOST)*, pages 67–74. IEEE, 2013.

[71] Yaguang Li, Yishuang Lin, Meghna Madhusudan, Arvind Sharma, Wenbin Xu, Sachin S Sapatnekar, Ramesh Harjani, and Jiang Hu. A customized graph neural network model for guiding analog ic placement. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2020.

[72] Rongjian Liang, Hua Xiang, Diwesh Pandey, Lakshmi Reddy, Shyam Ramji, Gi-Joon Nam, and Jiang Hu. Drc hotspot prediction at sub-10nm process nodes using customized convolutional network. In *Proceedings of the 2020 International Symposium on Physical Design*, pages 135–142, 2020.

[73] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[74] Derek Lim, Joshua Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. *International Conference on Learning Representations*, 2022.

[75] Ting-Ru Lin, Yunfan Li, Massoud Pedram, and Lizhong Chen. Design space exploration of memory controller placement in throughput processors with deep learning. *IEEE Computer Architecture Letters*, 18(1):51–54, 2019.

[76] Zhe Lin, Jieru Zhao, Sharad Sinha, and Wei Zhang. Hl-pow: A learning-based power modeling framework for high-level synthesis. In *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 574–580. IEEE, 2020.

[77] Mingjie Liu, Walker J Turner, George F Kokai, Brucek Khailany, David Z Pan, and Haoxing Ren. Parasitic-aware analog circuit sizing with graph neural networks and bayesian optimization. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1372–1377. IEEE, 2021.

[78] Shikun Liu, Tianchun Li, Yongbin Feng, Nhan Tran, Han Zhao, Qiang Qiu, and Pan Li. Structural re-weighting improves graph domain adaptation. In *International Conference on Machine Learning*, pages 21778–21793. PMLR, 2023.

[79] Shikun Liu, Deyu Zou, Han Zhao, and Pan Li. Pairwise alignment improves graph domain adaptation. *International Conference on Machine Learning*, 2024.

[80] Shuhan Liu and Kaize Ding. Beyond generalization: A survey of out-of-distribution adaptation on graphs. *arXiv preprint arXiv:2402.11153*, 2024.

[81] Daniel Lo, Taejoon Song, and G Edward Suh. Prediction-guided performance-energy tradeoff for interactive applications. In *Proceedings of the 48th International Symposium on Microarchitecture*, pages 508–520. ACM, 2015.

[82] Yi-Chen Lu, Siddhartha Nath, Sai Pentapati, and Sung Kyu Lim. Eco-gnn: Signoff power prediction using graph neural networks with subgraph approximation. *ACM Transactions on Design Automation of Electronic Systems*, 28(4):1–22, 2023.

[83] Yi-Chen Lu, Haoxing Ren, Hao-Hsiang Hsiao, and Sung Kyu Lim. Gan-place: Advancing open source placers to commercial-quality using generative adversarial networks and transfer learning. *ACM Transactions on Design Automation of Electronic Systems*, 29(2):1–17, 2024.

[84] Yi Ma, Jianye Hao, Yaodong Yang, Han Li, Junqi Jin, and Guangyong Chen. Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990*, 2019.

[85] Yuzhe Ma, Haoxing Ren, Brucek Khailany, Harbinder Sikka, Lijuan Luo, Karthikeyan Natarajan, and Bei Yu. High performance graph convolutional networks with applications in testability analysis. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[86] Farzaneh Mahdisoltani, Ioan Stefanovici, and Bianca Schroeder. Proactive error prediction to improve storage system reliability. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 391–402, 2017.

[87] Alireza Mahzoon, Daniel Große, and Rolf Drechsler. Revsca: Using reverse engineering to bring light into backward rewriting for big and dirty multipliers. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[88] Hosein Mohammadi Makrani, Farnoud Farahmand, Hossein Sayadi, Sara Bondi, Sai Manoj Pudukotai Dinakarrao, Houman Homayoun, and Setareh Rafatirad. Pyramid: Machine learning framework to estimate the optimal timing and resource usage of a high-level synthesis design. In *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, pages 397–403. IEEE, 2019.

[89] Charith Mendis, Alex Renda, Saman Amarasinghe, and Michael Carbin. Ithemal: Accurate, portable and fast basic block throughput estimation using deep neural networks. In *International Conference on machine learning*, pages 4505–4515. PMLR, 2019.

[90] Erxue Min, Runfa Chen, Yatao Bian, Tingyang Xu, Kangfei Zhao, Wenbing Huang, Peilin Zhao, Junzhou Huang, Sophia Ananiadou, and Yu Rong. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.

[91] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[92] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. Dag-aware aig rewriting a fresh look at combinational logic synthesis. In *Proceedings of the 43rd annual Design Automation Conference*, pages 532–535, 2006.

[93] Nikita Mishra, Connor Imes, John D Lafferty, and Henry Hoffmann. Caloree: Learning control for predictable latency and low energy. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 184–198, 2018.

[94] Federico Monti, Karl Otness, and Michael M Bronstein. Motifnet: a motif-based graph convolutional network for directed graphs. In *2018 IEEE data science workshop (DSW)*, pages 225–228. IEEE, 2018.

[95] Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research*, 6(May):783–816, 2005.

[96] Audrey Olivier, Michael D Shields, and Lori Graham-Brady. Bayesian neural networks for uncertainty quantification in data-driven materials modeling. *Computer methods in applied mechanics and engineering*, 386:114079, 2021.

[97] Kenneth O'Neal, Philip Brisk, Emily Shriver, and Michael Kishinevsky. Halwpe: Hardware-assisted light weight performance estimation for gpus. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2017.

[98] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[99] Ashutosh Pattnaik, Xulong Tang, Adwait Jog, Onur Kayiran, Asit K Mishra, Mahmut T Kandemir, Onur Mutlu, and Chita R Das. Scheduling techniques for gpu architectures with processing-in-memory capabilities. In *Proceedings of the 2016 International Conference on Parallel Architectures and Compilation*, pages 31–44, 2016.

[100] Mangpo Phothilimthana, Sami Abu-El-Haija, Kaidi Cao, Bahare Fatemi, Michael Burrows, Charith Mendis, and Bryan Perozzi. Tpugraphs: A performance prediction dataset on large tensor computational graphs. *Advances in Neural Information Processing Systems*, 36, 2023.

[101] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[102] Haoxing Ren, George F Kokai, Walker J Turner, and Ting-Sheng Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[103] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Computing Surveys (CSUR)*, 54(4):1–34, 2021.

[104] Emanuele Rossi, Bertrand Charpentier, Francesco Di Giovanni, Fabrizio Frasca, Stephan Günnemann, and Michael M Bronstein. Edge directionality improves learning on heterophilic graphs. In *Learning on Graphs Conference*, pages 25–1. PMLR, 2024.

[105] Boshen Shi, Yongqing Wang, Fangda Guo, Bingbing Xu, Huawei Shen, and Xueqi Cheng. Graph domain adaptation: Challenges, progress and prospects. *arXiv preprint arXiv:2402.00904*, 2024.

[106] Huihong Shi, Haoran You, Yang Zhao, Zhongfeng Wang, and Yingyan Lin. Nasa: Neural architecture search and acceleration for hardware inspired hybrid networks. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pages 1–9, 2022.

[107] Aebel Joe Shibu, Shilpa N, and Pratyush Kumar. Verlpy: Python library for verification of digital designs with reinforcement learning. In *Proceedings of the First International Conference on AI-ML Systems*, pages 1–7, 2021.

[108] Brett Shook, Prateek Bhansali, Chandramouli Kashyap, Chirayu Amin, and Siddhartha Joshi. Mlparest: Machine learning based parasitic estimation for custom circuit design. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[109] MA Shubin. Discrete magnetic laplacian. *Communications in mathematical physics*, 164(2):259–275, 1994.

[110] Rahul Singh, Abhishek Chakraborty, and BS Manoj. Graph fourier transform based on directed laplacian. In *2016 International Conference on Signal Processing and Communications (SPCOM)*, pages 1–5. IEEE, 2016.

[111] IEEE Electronics Packaging Society. Heterogeneous integration roadmap. `https://eps.ieee.org/technology/heterogeneous-integration-roadmap.html`.

[112] Pramod Subramanyan, Nestan Tsiskaridze, Wenchao Li, Adria Gascón, Wei Yang Tan, Ashish Tiwari, Natarajan Shankar, Sanjit A Seshia, and Sharad Malik. Reverse engineering digital circuits using structural and functional analyses. *IEEE Transactions on Emerging Topics in Computing*, 2(1):63–80, 2013.

[113] Amit Surana, Soundar Kumara*, Mark Greaves, and Usha Nandini Raghavan. Supply-chain networks: a complex adaptive systems perspective. *International Journal of Production Research*, 43(20):4235–4265, 2005.

[114] Ondřej Sỳkora, Phitchaya Mangpo Phothilimthana, Charith Mendis, and Amir Yazdanbakhsh. Granite: A graph neural network model for basic block throughput estimation. In *2022 IEEE International Symposium on Workload Characterization (IISWC)*, pages 14–26. IEEE, 2022.

[115] Aysa Fakheri Tabrizi, Logan Rakai, Nima Karimpour Darav, Ismail Bustany, Laleh Behjat, Shuchang Xu, and Andrew Kennings. A machine learning framework to identify detailed routing short violations from a placed netlist. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2018.

[116] Veronika Thost and Jie Chen. Directed acyclic graph neural networks. In *International Conference on Learning Representations*, 2020.

[117] Aviral Kumar Tiwari, Micheal Kofi Boachie, and Rangan Gupta. Network analysis of economic and financial uncertainties in advanced economies: Evidence from graph-theory. 2021.

[118] Zekun Tong, Yuxuan Liang, Changsheng Sun, Xinke Li, David Rosenblum, and Andrew Lim. Digraph inception convolutional networks. *Advances in neural information processing systems*, 33:17907–17918, 2020.

[119] Zekun Tong, Yuxuan Liang, Changsheng Sun, David S Rosenblum, and Andrew Lim. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970*, 2020.

[120] Ecenur Ustun, Chenhui Deng, Debjit Pal, Zhijing Li, and Zhiru Zhang. Accurate operation delay prediction for fpga hls using graph neural networks. In *Proceedings of the 39th International Conference on Computer-Aided Design*, pages 1–9, 2020.

[121] Shobha Vasudevan, Wenjie Joe Jiang, David Bieber, Rishabh Singh, C Richard Ho, Charles Sutton, et al. Learning semantic representations to verify hardware designs. *Advances in Neural Information Processing Systems*, 34, 2021.

[122] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[123] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *International Conference on Learning Representations*, 2018.

[124] Andre Vladimirescu. *The SPICE book*. John Wiley & Sons, Inc., 1994.

[125] Stephan M Wagner and Nikrouz Neshat. Assessing the vulnerability of supply chains using graph theory. *International journal of production economics*, 126(1):121–129, 2010.

[126] Hanrui Wang, Kuan Wang, Jiacheng Yang, Linxiao Shen, Nan Sun, Hae-Seung Lee, and Song Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[127] Haorui Wang, Haoteng Yin, Muhan Zhang, and Pan Li. Equivariant and stable positional encoding for more powerful graph neural networks. *International Conference on Learning Representations*, 2022.

[128] Haoyu Peter Wang, Nan Wu, Hang Yang, Cong Hao, and Pan Li. Unsupervised learning for combinatorial optimization with principled objective relaxation. *Advances in Neural Information Processing Systems*, 35:31444–31458, 2022.

[129] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. *EMNLP*, 2021.

[130] Shuhei Watanabe. Tree-structured parzen estimator: Understanding its algorithm components and their roles for better empirical performance. *arXiv preprint arXiv:2304.11127*, 2023.

[131] Wei Wen, Hanxiao Liu, Yiran Chen, Hai Li, Gabriel Bender, and Pieter-Jan Kindermans. Neural predictor for neural architecture search. In *European conference on computer vision*, pages 660–676. Springer, 2020.

[132] Steven Euijong Whang, Yuji Roh, Hwanjun Song, and Jae-Gil Lee. Data collection and quality challenges in deep learning: A data-centric ai perspective. *The VLDB Journal*, 32(4):791–813, 2023.

[133] Nan Wu, Jiwon Lee, Yuan Xie, and Cong Hao. Lostin: Logic optimization via spatio-temporal information with hybrid graph models. In *2022 IEEE 33rd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pages 11–18. IEEE, 2022.

[134] Nan Wu, Yingjie Li, Cong Hao, Steve Dai, Cunxi Yu, and Yuan Xie. Gamora: Graph learning based symbolic reasoning for large-scale boolean networks. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

[135] Nan Wu and Yuan Xie. A survey of machine learning for computer architecture and systems. *ACM Computing Surveys (CSUR)*, 55(3):1–39, 2022.

[136] Nan Wu, Yuan Xie, and Cong Hao. Ironman: Gnn-assisted design space exploration in high-level synthesis via reinforcement learning. In *Proceedings of the 2021 on Great Lakes Symposium on VLSI*, pages 39–44, 2021.

[137] Nan Wu, Hang Yang, Yuan Xie, Pan Li, and Cong Hao. High-level synthesis performance prediction using gnns: Benchmarking, modeling, and advancing. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pages 49–54, 2022.

[138] Renzhi Wu, Sanya Chaba, Saurabh Sawlani, Xu Chu, and Saravanan Thirumuruganathan. Zeroer: Entity resolution using zero labeled examples. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1149–1164, 2020.

[139] Renzhi Wu, Prem Sakala, Peng Li, Xu Chu, and Yeye He. Demonstration of panda: a weakly supervised entity matching system. *Proceedings of the VLDB Endowment*, 2021.

[140] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *International Conference on Learning Representations*, 2019.

[141] Xiaoqing Xu, Nishi Shah, Andrew Evans, Saurabh Sinha, Brian Cline, and Greg Yeric. Standard cell library design and optimization methodology for asap7 pdk. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 999–1004. IEEE, 2017.

[142] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, et al. Improving service availability of cloud systems by predicting disk error. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*, pages 481–494, 2018.

[143] Jiang Xun, Zhuomin Chai, Yuxiang Zhao, Yibo Lin, Runsheng Wang, and Ru Huang. Circuitnet 2.0: An advanced dataset for promoting machine learning innovations in realistic chip design environment. In *The Twelfth International Conference on Learning Representations*, 2024.

[144] Hanchen Ye, Hyegang Jun, and Deming Chen. Hida: A hierarchical dataflow compiler for high-level synthesis. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 1*, pages 215–230, 2024.

[145] 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? *Advances in neural information processing systems*, 34:28877–28888, 2021.

[146] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. Developing synthesis flows without human knowledge. In *Proceedings of the 55th Annual Design Automation Conference*, pages 1–6, 2018.

[147] Cunxi Yu and Wang Zhou. Decision making in synthesis cross technologies using lstms and transfer learning. In *Proceedings of the 2020 ACM/IEEE Workshop on Machine Learning for CAD*, pages 55–60, 2020.

[148] Shih-Yuan Yu, Rozhin Yasaei, Qingrong Zhou, Tommy Nguyen, and Mohammad Abdullah Al Faruque. Hw2vec: A graph learning tool for automating hardware security. In *2021 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 13–23. IEEE, 2021.

[149] Guo Zhang, Hao He, and Dina Katabi. Circuit-gnn: Graph neural networks for distributed circuit design. In *International conference on machine learning*, pages 7364–7373. PMLR, 2019.

[150] Li Lyna Zhang, Shihao Han, Jianyu Wei, Ningxin Zheng, Ting Cao, Yuqing Yang, and Yunxin Liu. Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, pages 81–93, 2021.

[151] Muhan Zhang, Shali Jiang, Zhicheng Cui, Roman Garnett, and Yixin Chen. D-vae: A variational autoencoder for directed acyclic graphs. *Advances in neural information processing systems*, 32, 2019.

[152] Xitong Zhang, Yixuan He, Nathan Brugnone, Michael Perlmutter, and Matthew Hirn. Magnet: A neural network for directed graphs. *Advances in neural information processing systems*, 34:27003–27015, 2021.

[153] Guangwei Zhao and Kaveh Shamsi. Graph neural network based netlist operator detection under circuit rewriting. In *Proceedings of the Great Lakes Symposium on VLSI 2022*, pages 53–58, 2022.

[154] Jieru Zhao, Liang Feng, Sharad Sinha, Wei Zhang, Yun Liang, and Bingsheng He. Comba: A comprehensive model-based analysis framework for high level synthesis of real applications. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 430–437. IEEE, 2017.

[155] Jieru Zhao, Tingyuan Liang, Sharad Sinha, and Wei Zhang. Machine learning based routing congestion prediction in fpga high-level synthesis. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 1130–1135. IEEE, 2019.

[156] Xinnian Zheng, Lizy K John, and Andreas Gerstlauer. Accurate phase-level cross-platform power and performance estimation. In *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2016.

[157] Yunxing Zuo, Mingde Qin, Chi Chen, Weike Ye, Xiangguo Li, Jian Luo, and Shyue Ping Ong. Accelerating materials discovery with bayesian optimization and graph deep learning. *Materials Today*, 51:126–135, 2021.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments (e.g. for benchmarks)...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Section D
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] See Section D
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]