GNN-ACLP: Graph Neural Networks based Analog Circuit Link Prediction

Guanyuan Pan[a], Tiansheng Zhou[b], Bingtao Ma[b], Yaqi Wang[c], Jianxiang Zhao[b], Zhi Li[b], Yugui Lin[d], Pietro Liò[e], Shuai Wang[b,*]

[a]*HDU-ITMO Joint Institute, Hangzhou Dianzi University, No. 1158, 2nd Avenue, Xiasha Higher Education Zone, Jianggan District, Hangzhou, 310018, Zhejiang Province, China*

[b]*Intelligent Information Processing Laboratory, Hangzhou Dianzi University, No. 1158, 2nd Avenue, Xiasha Higher Education Zone, Jianggan District, Hangzhou, 310018, Zhejiang Province, China*

[c]*College of Media Engineering, Communication University of Zhejiang, No. 998, Xueyuan Street, Xiasha Higher Education Zone, Jianggan District, Hangzhou, 310018, Zhejiang Province, China*

[d]*School of Computer Science and Technology, South China Business College, Guangdong University of Foreign Studies, 181 Liangtian Middle Road, Baiyun District, Guangzhou, 510545, Guangdong Province, China*

[e]*Department of Computer Science and Technology, University of Cambridge, William Gates Building, 15 JJ Thomson Avenue, Cambridge, CB3 0FD, Cambridgeshire, England, United Kingdom*

Abstract

Circuit link prediction identifying missing component connections from incomplete netlists is crucial in automating analog circuit design. However, existing methods face three main challenges: 1) Insufficient use of topological patterns in circuit graphs reduces prediction accuracy; 2) Data scarcity due to the complexity of annotations hinders model generalization; 3) Limited adaptability to various netlist formats. We propose GNN-ACLP, a Graph Neural Networks (GNNs) based framework featuring three innovations to tackle these challenges. First, we introduce the SEAL (Subgraphs, Em-

---

*Corresponding author

Email addresses: panguanyuan@hdu.edu.cn (Guanyuan Pan), zhoutiansheng_2024@163.com (Tiansheng Zhou), mabingtao@hdu.edu.cn (Bingtao Ma), wangyaqi@cuz.edu.cn (Yaqi Wang), 246270147@hdu.edu.cn (Jianxiang Zhao), lizhi2513@hdu.edu.cn (Zhi Li), yuguilin0209@163.com (Yugui Lin), pl219@cam.ac.uk (Pietro Liò), shuaiwang@hdu.edu.cn (Shuai Wang)

beddings, and Attributes for Link Prediction) framework and achieve port-level accuracy in circuit link prediction. Second, we propose Netlist Babel Fish, a netlist format conversion tool leveraging retrieval-augmented generation (RAG) with a large language model (LLM) to enhance the compatibility of netlist formats. Finally, we construct SpiceNetlist, a comprehensive dataset that contains 775 annotated circuits across 10 different component classes. Experimental results achieve accuracy improvements of 16.08% on SpiceNetlist, 11.38% on Image2Net, and 16.01% on Masala-CHAI in intra-dataset evaluation, while maintaining accuracy from 92.05% to 99.07% in cross-dataset evaluation, exhibiting robust feature transfer capabilities.

Keywords: Circuit Link Prediction, GNNs, LLM, RAG, EDA

1. Introduction

The design of analog circuits has mainly been performed manually by experienced engineers. However, analog design is susceptible to layout geometry [1] and even experienced engineers face difficulties when designing them. Therefore, automating analog circuit design has gained significant attention recently. One main task of analog design automation is circuit link prediction, which involves inferring missing component interconnections from incomplete netlists.

There are two conventional approaches for general link prediction: heuristic methods and learning-based approaches. Heuristic approaches can be divided into two categories: local similarity metrics, which include the Jaccard index, common neighbors, preferential attachment, and resource allocation, and global similarity metrics, including the Katz index and Sim-Rank [2]. Although heuristic methods offer simplicity and interpretability through predefined topological features, they have limited generalizability due to their dependence on handcrafted structural assumptions about link formation mechanisms [3]. Therefore, they are not commonly used in circuit link prediction.

In the Electronic Design Automation (EDA) field, where circuit netlists represent graph-structured data, machine learning (ML) approaches have gained more prominence. Genssler et al. proposed a novel method using brain-inspired hyperdimensional computing (HDC) for encoding and recognizing gate-level netlists [4]. However, the high-dimensional vector operations involved in HDC led to significant computational and storage over-

2

head, especially when dealing with large-scale graph data [5]. Luo et al. proposed a novel neural model called Directional Equivariant Hypergraph Neural Network (DE-HNN) for effective representation learning on directed hypergraphs, particularly for circuit netlists in chip design [6]. However, DE-HNN performing hypergraph diffusion or tensor operations could exponentially increase computational costs as the hyperedge order increases, making it challenging to scale the approach for large chip design scenarios [7].

GNNs can learn from circuit netlists' inherent graph structure, making them well-suited for circuit link prediction. However, a critical challenge lies in the lack of annotated training data caused by high human workload and data sensitivity [8]. Furthermore, the diversity of netlist formats poses significant challenges in analog design automation. Additionally, customized netlist formats are often created by modifying standard formats. Consequently, different datasets are often created in different formats, complicating the training processes for machine learning methods.

To address these challenges, we propose **GNN-ACLP**, a GNNs-based method for analog circuit link prediction, as illustrated in Figure 1, and SpiceNetlist, a large-scale netlist dataset whose statistics are demonstrated in Table 1. **1)** On the one hand, we formulate circuit link prediction as an undirected graph link prediction problem, where nodes represent circuit components and edges denote connections. To solve this task, we leverage the SEAL framework, trained using DRNL one-hot encoding augmented with original component node features for enhanced representation learning. **2)** On the other hand, we introduce **Netlist Babel Fish**, a framework featuring RAG for LLM-based interpretation and enabling bidirectional netlist format conversion through domain-specific knowledge integration, to overcome the challenge of differing dataset formats. **3)** Finally, we develop a robust preprocessing pipeline to detect and correct content errors and syntactic inconsistencies across SPICE netlist datasets. We then apply the pipeline on two small-scale datasets and integrate them into one unified large-scale dataset, SpiceNetlist, to support the training and evaluation of circuit link prediction methods. SpiceNetlist has 775 annotated circuits in both SPICE and JSON formats, featuring 10 component types. The experimental results demonstrate significant performance gains in intra-dataset evaluation, achieving accuracy improvements of 16.08% on SpiceNetlist, 11.38% on Image2Net, and 16.01% on Masala-CHAI compared to the baseline approach. Furthermore, our method maintains accuracy from 92.05% to 99.07% in cross-dataset evaluation, exhibiting robust feature transfer capabilities.

3

The contributions of our work are outlined below:

- **GNN-ACLP**, a port-level accurate circuit link prediction method based on the SEAL framework using graph neural networks (GNNs). To the best of our knowledge, this is the first work to address circuit link prediction at the port level, as opposed to the conventional component-level approach.

- **Netlist Babel Fish**, a netlist format converter enabling port-level netlist parsing across multiple formats, leveraging RAG for LLM-based interpretation.

- **SpiceNetlist**, a novel circuit netlist dataset comprising 775 annotated circuits in both SPICE and JSON formats, featuring 10 component types to facilitate the training and evaluation of circuit link prediction methods.

Table 1: SpiceNetlist statistic.

Dataset	Graphs	Classes	Avg. Nodes	Avg. Edges
SpiceNetlist	775	10	13.78	15.60

2. Preliminaries and problem formulation

The circuit link prediction problem is a part of the circuit completion problem. One can define the circuit completion problem as follows:

Problem 2.1 (circuit completion problem) Let v_0 and v_1 represent the netlists of two circuit schematics, where v_1 is a partial copy of v_0 missing all connections associated with a component x that are present in v_0. The circuit completion problem involves predicting the type of the missing component x and all of its missing connections, where $x \in \{1, 2, 3, \ldots, k\}$.

Here, netlists define the connections between circuit components, including ports, each with a unique ID and type [9, 10].

There are several challenges in solving circuit completion problems. First, the problem implicitly assumes an explicit criterion for determining a circuit's validity, which is necessary for verifying a solution's correctness. Although some checks can be performed, no definitive criteria exist to validate a circuit
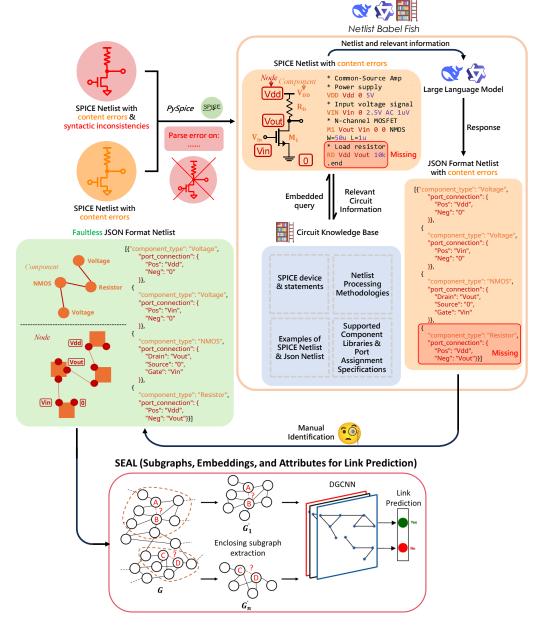
Figure 1: Illustration of GNN-ACLP. We first preprocess SPICE netlist datasets by correcting errors and syntactic inconsistencies, generating faultless JSON netlists. We then enable bidirectional format conversion via Netlist Babel Fish, a RAG-based LLM framework plus domain-specific knowledge. The netlists are then modeled as graphs, and we apply SEAL for circuit link prediction.
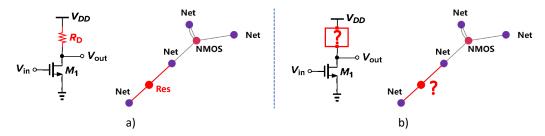
Figure 2: Illustration of the circuit link prediction problem. **a)** A schematic netlist representation, **b)** Schematic netlist representation with a missing component.

netlist. Second, it may not be relevant to one's interests even if a circuit is valid. As long as both v_0 and v'_0 are netlists of designs that are considered "interesting" (where "interesting" is defined by the specific application), it may be acceptable to predict either as a valid completion of the partial netlist v_1. However, v'_0 may be a trivial completion irrelevant to the particular application being studied [11, 12, 13]. To address these challenges, we abandon any auxiliary information from the netlists and convert each netlist into an undirected graph, denoted as $G = (V, E, \phi)$. In this graph, the set of vertices V represents the ports of all components in the netlist, while the edges denote the connections between these ports in E that link the corresponding vertices. Additionally, each vertex v has an integer type, denoted as $\phi(v)$, taken from the set $\{1, 2, 3, \ldots, k\}$.

We then define the circuit link prediction problem:

Problem 2.2 (circuit link completion problem on graphs) Let $G = (V, E, \phi)$ be a graph of a netlist with $\phi(v) \in [k]$ where $[k] := \{1, 2, 3, \ldots, k\}$. Let G' be a graph obtained by removing an arbitrary vertex $u \in V$ from G. Given the value of $\phi(u)$ and G', compute the set of neighbors of u in G.

Here, the set of neighbors of a node u is defined as the following set: $\{v \in V : (u, v) \in E\}$. We propose data-driven solutions to address the problem. Specifically, we learn a map, $\hat{\xi} : G, [k] \to 2^V$, that takes as input a graph G and a component type $\phi \in [k]$. This mapping returns a subset of vertices (the power set of vertices V, represented as 2^V) that indicate the connections for the ports of components within graph G. We provide a visual illustration of this problem in Figure 2 and 3.
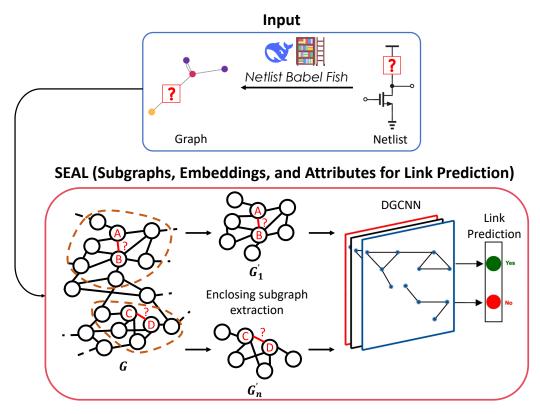
Figure 3: Architecture of our circuit link prediction architecture with SEAL. Netlists are converted into graph representations and processed by SEAL, which extracts and encloses subgraphs to generate training samples for GNN-based link prediction. We implement DGCNN as the GNN architecture.

3. Link prediction through GNNs

In Problem 2.2, we are given a graph G containing various node types and a distinguished node v. The objective of the problem is to predict the neighborhood of v within G. This problem can be simplified by determining whether an edge should exist between v and each neighboring node u in G. The goal is to assess whether two nodes in the graph will likely be connected by an edge based on the other nodes' connectivity. Therefore, this connection test can be framed as a link prediction problem.

We select the SEAL framework [14], which we illustrate in Figure 3. The idea is to enclose subgraphs around the links in a graph representing positive class instances to extract training data. For a pair of nodes (x, y), an enclosing subgraph is defined as a subgraph that includes the h-hop neighborhood of both x and y. In practice, this method is effective for the link prediction task. This work proposes a node labeling approach denoted as Double-Radius Node Labeling (DRNL) to label nodes within the subgraphs. DRNL aims to identify the different roles of nodes while preserving structural information. We adopt the SEAL framework for link prediction in circuit graphs and train it using DRNL one-hot encoding combined with the original component node features.

It is important to note that our work may not address the circuit link prediction problem on graphs since there can be multiple valid ways to link ports in a partial circuit [15, 16, 17]. Therefore, the proposed method may occasionally encounter failures. However, we show through extensive experiments that graph machine-learning methods yield reliable results on real-world datasets to support human experts in the design, synthesis, and evaluation processes.

4. Netlist Babel Fish: Netlist Format Converter

To address the challenge of differing dataset formats, we develop Netlist Babel Fish, a framework that enables bidirectional conversion between netlist formats through integration with domain-specific information. Netlist Babel Fish integrates an LLM with a RAG system grounded in a structured SPICE knowledge base. This knowledge base includes brief references of the SPICE devices and statements [18], methodologies for the netlist process, supported component libraries with port assignment specifications, and examples illustrating practical implementations. We use DeepSeek-V2.5 [19, 20], DeepSeek-V3-0324 [21, 22] and Qwen3-30B-A3B [23] as the LLM here. We show the

workflow of Netlist Babel Fish converting a SPICE netlist to a custom JSON format [24] in Figure 4. This workflow is also reversible for conversion from the custom JSON format to SPICE.

While the current implementation specifically addresses SPICE and specific JSON format conversions, the modular architecture theoretically supports arbitrary netlist format transformations given corresponding domain knowledge bases. The system's extensibility derives from its decoupled knowledge representation framework, where format-specific conversion rules are explicitly encoded rather than implicitly learned.

5. SpiceNetlist: Datasets Rectification

For the circuit-link-prediction problem, there are very few datasets available, and the existing ones do not contain enough readily accessible netlists for practical training and evaluation. Moreover, existing SPICE netlist datasets often contain content errors [24] and syntactic inconsistencies, leading to their incompatibility with Netlist Babel Fish for proper parsing.

To mitigate this issue, we employ PySpice [25] to automatically detect and filter out SPICE netlists with syntactic inconsistencies. Subsequently, we convert all datasets to JSON format with Netlist Babel Fish, and perform manual verification to identify and rectify netlists with content errors in both formats. We illustrate this preprocessing pipeline in Figure 5.

Following this pipeline, we integrate two datasets — Kicad Github [26] and AMSNet [27] as one unified large-scale dataset, SpiceNetlist. Table 1 shows SpiceNetlist's statistics. We also apply the preprocessing pipeline to three additional datasets: Image2Net[1], Masala-CHAI [29] and AnalogGenie [30].

6. Experiments

6.1. Dataset Processing

For our experiments, we use SpiceNetlist, Image2Net, Masala-CHAI [29] and AnalogGenie[2] [30]. Table 2 shows the statistics of all datasets, and table

---

[1]Image2Net is provided by Yiren Pan (panyiren@hdu.edu.cn), who is currently writing a paper about his findings, and originated from the 2024 China Postgraduate IC Innovation Competition - EDA Elite Challenge Contest [28] Image Dataset.

[2]Given the substantial size of AnalogGenie, we randomly select a representative 10% subset for our experiments.

Netlist Babel Fish

Large Language Model

Netlist and relevant information

SPICE Netlist

Customized JSON Format Netlist

```
* Common-Source Amp
* Power supply
VDD Vdd 0 5V
* Input voltage signal
VIN Vin 0 2.5V AC 1uV
* N-channel MOSFET
M1 Vout Vin 0 0 NMOS
W=50u L=1u
* Load resistor
RD Vdd Vout 10k
.end
```

Node Component

Vdd V_{DD}

R_D

Vout

V_{In} M_1

Vin   0

Component

Voltage

NMOS   Resistor

Voltage

Node

Vdd

Vout

Vin   0

```
[{"component_type": "Voltage",
   "port_connection": {
      "Pos": "Vdd",
      "Neg": "0"
}},
{
   "component_type": "Voltage",
   "port_connection": {
      "Pos": "Vin",
      "Neg": "0"
}},
{
   "component_type": "NMOS",
   "port_connection": {
      "Drain": "Vout",
      "Source": "0",
      "Gate": "Vin"
}},
{
   "component_type": "Resistor",
   "port_connection": {
      "Pos": "Vdd",
      "Neg": "Vout"}}]
```

Embedded query

Relevant Circuit Information

Circuit Knowledge Base

C device - Capacitor.
C{name} {+node} {-node}
[{model}] {value} [IC={initial}]
Examples:
CLOAD 15 0 20pF
CFDBK 3 33 CMOD 10pF
IC=1.5v
......

1. Component Identification
......
2. Netlist Parsing Guidelines
......
3. PMOS/NMOS Identification
......
4. PNP/NPN Transistor Identification

SPICE device & statements

Netlist Processing Methodologies

```
SPICE:
'''
Q3 (Vout VB \-5V 0) npn
......
'''
json dict:
'''
[{'component_type': 'NPN',
    'port_connection': {'Base': 'VB',
            'Collector': 'Vout',
            'Emitter': '\\-5V'}},
...
    .....]
'''
```

| Component Category | Subcategory | Ports |
......
| Voltage | Voltage | Pos, Neg |
| Current | Current | In, Out |
......
Important Notes:
Ensure correct port assignments. The `component_type` (corresponding to Component Category) must not exceed the defined scope!

Examples of SPICE Netlist & Json Netlist

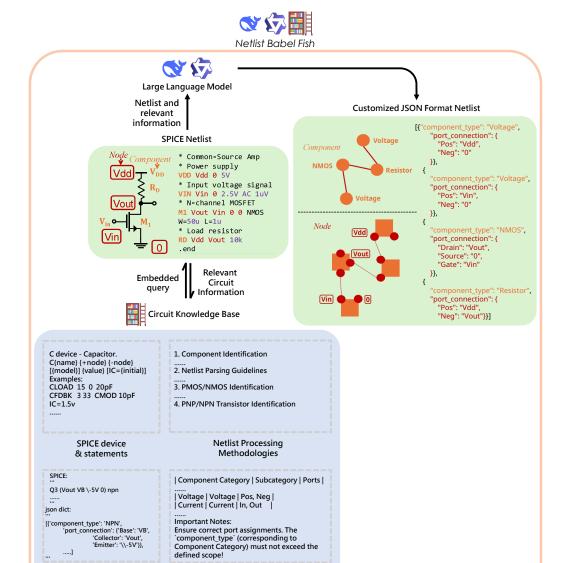**Supported Component Libraries & Port Assignment Specifications**

Figure 4: Workflow of Netlist Babel Fish. A SPICE netlist is first processed by an embedded query system that retrieves relevant information from our circuit knowledge base, which includes netlist preprocessing methodologies, supported component libraries including port assignment specifications, and exemplar netlists in both SPICE and JSON formats. The netlist and retrieved data are then fed into an LLM, which generates the corresponding JSON netlist as output.
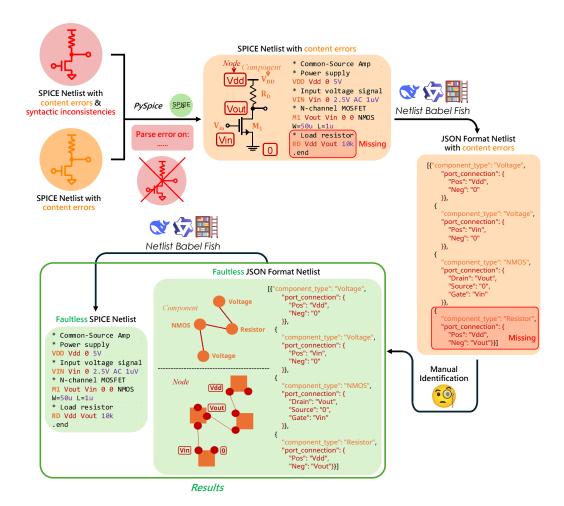
Figure 5: SPICE netlist datasets preprocessing pipeline. SPICE netlists are first validated using PySpice, with syntactically inconsistent netlists being filtered out. The remaining netlists undergo conversion to JSON format via Netlist Babel Fish, where content errors are manually corrected. The resulting faultless JSON netlists are then processed back through Netlist Babel Fish to generate faultless SPICE equivalents.

11

3 lists all components featured by all datasets.

Table 2: Statistics of all datasets.

Dataset	Graphs	Classes	Avg. Nodes	Avg. Edges
SpiceNetlist	775	10	13.78	15.60
Image2Net	1960	13	26.36	37.80
Masala-CHAI	4057	13	15.72	19.97
AnalogGenie	4064	9	63.89	168.73

Table 3: Component types and their labels/ports in the datasets.

Component	Label	Ports
PMOS/NMOS	PMOS/NMOS	Drain, Source, Gate
Voltage Source	Voltage	Pos, Neg
Current Source	Current	In, Out
BJT (NPN/PNP)	NPN/NPN_cross/PNP/PNP_cross	Base, Emitter, Collector
Diode	Diode	In, Out
DISO Amplifier	Diso_amp	InN, InP, Out
SISO Amplifier	Siso_amp	In, Out
DIDO Amplifier	Dido_amp	InN, InP, OutN, OutP
Passive Components (Cap, Ind, Res)	Cap, Ind, Res	Pos, Neg

We opt for an approach similar to batching in the graph learning domain, where we stack the adjacency matrices in diagonal blocks to represent the entire dataset as a single huge graph. Let $A_i \in [0,1]^{N_i \times N_i}$ be the adjacency matrix of graph G_i with N_i nodes. Then, the resulting graph will have $N := \sum_{1 \leq j \leq n} N_j$ nodes. The stacked adjacency matrix A with dimensions, $N \times N$, and the corresponding vector with concatenated node features are defined as:

$$A = \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_n \end{bmatrix}, \quad X = \begin{bmatrix} X_1 \\ \vdots \\ X_n \end{bmatrix}$$

12

Link prediction methods typically require a single large graph as input to learn the network structure [31, 32]. Therefore, stacking all graphs in the training dataset allows us to utilize off-the-shelf GNN methods for prediction with minimal additional computational overhead. The large adjacency matrix can also be stored in a compressed format using sparse data structures.

6.2. Experimental Setup

We conduct two distinct evaluation paradigms: intra-dataset evaluation and cross-dataset evaluation. For intra-dataset evaluation, models are trained and tested on the same dataset, specifically SpiceNetlist, Image2Net and Masala-CHAI [29]. For cross-dataset evaluation to assess the generalization capability of our approach, we first examine models trained on one dataset and evaluated on another from these three datasets, followed by additional validation using AnalogGenie [30].

For both evaluation paradigms, we employ two experimental configurations: one using the conventional dataset splitting strategy (train, validation, and test splits), and another employing the 5-fold cross-validation strategy, given the prevalent challenge of limited sizes in existing datasets. Here, the traditional split serves as an ablation study to assess the impact of 5-fold cross-validation, ensuring robustness in performance evaluation. For both configurations, we maintain a fixed data split ratio of 70% training, 20% test, and 10% validation. We also conduct experiments with the approach of [26] as the baseline, which uses the conventional dataset splitting strategy.

We adopt the SEAL framework with the following Pytorch Geometric implementation: batch size $= 1$, 2-hop neighborhoods and maximum 50 training epochs with early stopping. The early stopping criterion only engages once test accuracy shows improvement and exceeds 0.5000. Thereafter, training terminates if no improvement ≥ 0.0001 is observed for three consecutive epochs. We use the one-hot encoding of node labels generated through DRNL and concatenate them with the one-hot encoding of the component types. For SpiceNetlist, we set the learning rate to $1e - 6$. For Image2Net and Masala-CHAI, we set the learning rate to $1e - 6$ for the baseline and $6e - 8$ for our approach.

The training graph is constructed using the adjacency stacking method from Section 6.1. During each experiment, we remove one random vertex from each test graph and predict its connections. We repeat the experiment 10 times and report the average accuracy and ROC-AUC.

6.3. Results and Analysis

We present a comprehensive comparison of circuit link prediction performance between our work and the baseline [26] in Table 4-6 and Figure 6-8. The experimental results demonstrate substantial performance improvements across all evaluation scenarios:

In the intra-dataset evaluation, **1)** We achieve accuracy gains of 11.08% (conventional split) and 16.08% (5-fold cross-validation) on SpiceNetlist; **2)** We achieve accuracy gains of 9.18% (conventional split) and 11.38% (5-fold cross-validation) on Image2Net; **3)** We achieve accuracy gains of 13.89% (conventional split) and 16.02% (5-fold cross-validation) on Masala-CHAI.

In the cross-dataset evaluation, our work maintains accuracy ranging from 92.05% (trained on Image2Net, tested on SpiceNetlist) to 99.07% (trained on SpiceNetlist, tested on AnalogGenie), demonstrating its strong capability to transfer learned features across different circuit netlist datasets.

Our work obtains promising results, indicating that the proposed framework effectively learns netlist structures and can be utilized for component link prediction tasks.

Table 4: Accuracy in intra-dataset evaluation.

Dataset	Baseline	Conv. Split (Ours)	5-fold CV (Ours)
SpiceNetlist	77.96% ± 1.71%	89.04% ± 1.18%	**94.04% ± 0.70%**
Image2Net	85.72% ± 1.95%	94.90% ± 0.25%	**97.10% ± 0.24%**
Masala-CHAI	80.40% ± 0.92%	94.29% ± 0.28%	**96.42% ± 1.24%**

Table 5: ROC-AUC in intra-dataset evaluation.

Dataset	Baseline	Conv. Split (Ours)	5-fold CV (Ours)
SpiceNetlist	0.8774 ± 0.0056	0.9645 ± 0.0097	**0.9866 ± 0.0019**
Image2Net	0.9367 ± 0.0164	0.9926 ± 0.0004	**0.9979 ± 0.0003**
Masala-CHAI	0.8995 ± 0.0099	0.9892 ± 0.0009	**0.9959 ± 0.0005**

7. Conclusion

We propose GNN-ACLP, a novel graph neural networks based framework with SEAL framework integration and achieve port-level accuracy in analog
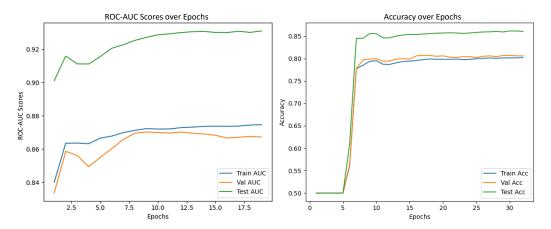
Figure 6: AUC and Accuracy variation tendencies observed in the baseline approach. Both indicators vibrate violently.
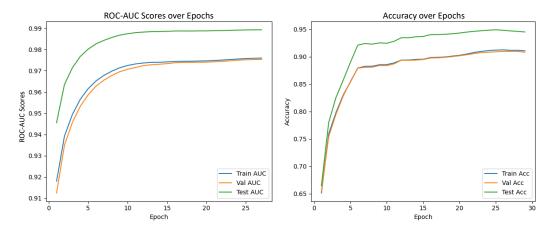

Figure 7: AUC and Accuracy variation tendencies observed in our approach with conventional dataset splitting strategy. Both indicators vibrate less violently than the baseline approach.
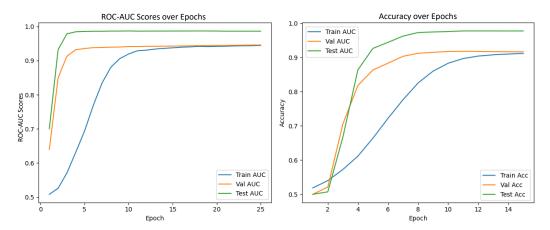
Figure 8: AUC and Accuracy variation tendencies observed in our approach with 5-fold cross-validation strategy. Both indicators increase steadily.

Table 6: Accuracy and ROC-AUC in cross-dataset evaluation.

Train Dataset	Test Dataset	Baseline Performance		5-fold CV (Ours) Performance	
		Accuracy	ROC-AUC	Accuracy	ROC-AUC
SpiceNetlist	Image2Net	$82.95\% \pm 1.45\%$	0.9163 ± 0.0114	$\mathbf{96.94\% \pm 0.88\%}$	$\mathbf{0.9963 \pm 0.0014}$
SpiceNetlist	Masala-CHAI	$77.00\% \pm 1.47\%$	0.8600 ± 0.0030	$\mathbf{95.98\% \pm 1.08\%}$	$\mathbf{0.9945 \pm 0.0017}$
Image2Net	SpiceNetlist	$75.10\% \pm 3.11\%$	0.8848 ± 0.0050	$\mathbf{92.05\% \pm 0.75\%}$	$\mathbf{0.9867 \pm 0.0021}$
Image2Net	Masala-CHAI	$75.47\% \pm 1.28\%$	0.8706 ± 0.0048	$\mathbf{96.12\% \pm 0.66\%}$	$\mathbf{0.9962 \pm 0.0007}$
Masala-CHAI	SpiceNetlist	$72.84\% \pm 8.50\%$	0.8678 ± 0.0424	$\mathbf{93.06\% \pm 1.66\%}$	$\mathbf{0.9859 \pm 0.0019}$
Masala-CHAI	Image2Net	$82.35\% \pm 8.01\%$	0.9293 ± 0.0151	$\mathbf{97.18\% \pm 0.85\%}$	$\mathbf{0.9974 \pm 0.0004}$
SpiceNetlist	AnalogGenie	$69.99\% \pm 2.17\%$	0.7822 ± 0.0330	$\mathbf{99.07\% \pm 0.82\%}$	$\mathbf{0.9995 \pm 0.0003}$
Image2Net	AnalogGenie	$71.23\% \pm 2.32\%$	0.7950 ± 0.0402	$\mathbf{98.73\% \pm 1.32\%}$	$\mathbf{0.9997 \pm 0.0002}$
Masala-CHAI	AnalogGenie	$74.19\% \pm 2.58\%$	0.8107 ± 0.0378	$\mathbf{98.68\% \pm 1.07\%}$	$\mathbf{0.9997 \pm 0.0002}$

circuit link prediction. We also propose Netlist Babel Fish, a netlist format conversion tool combining RAG with an LLM. Additionally, we create the SpiceNetlist dataset for training and evaluation of circuit link prediction methods. Results of the experiment show that our approach achieves state-of-the-art performance.

CRediT authorship contribution statement

Guanyuan Pan: Investigation, Validation, Software, Writing – original draft, Visualization. **Tiansheng Zhou**: Data curation, Visualization. **Bingtao Ma**: Writing – review and editing. **Yaqi Wang**: Conceptualization, Writing – review and editing. **Jianxiang Zhao**: Writing – review and editing. **Zhi Li**: Visualization. **Yugui Lin**: Validation. **Pietro Liò**: Writing – review and editing. **Shuai Wang**: Resources, Supervision, Writing – review and editing.

Data Availability

Data will be made available on request.

References

[1] H. Chen, M. Liu, X. Tang, K. Zhu, N. Sun, D. Z. Pan, Challenges and Opportunities toward Fully Automated Analog Layout Design, Journal of Semiconductors 41 (11) (2020) 111407. `doi:10.1088/1674-4926/41/11/111407`.
URL `https://dx.doi.org/10.1088/1674-4926/41/11/111407`

[2] A. Samad, M. Qadir, I. Nawaz, M. A. Islam, M. Aleem, A comprehensive survey of link prediction techniques for social network., EAI Endorsed Trans. Ind. Networks Intell. Syst. 7 (23) (2020) e3.

[3] D. Liben-Nowell, J. Kleinberg, The link prediction problem for social networks, in: Proceedings of the twelfth international conference on Information and knowledge management, 2003, pp. 556–559.

[4] P. R. Genssler, L. Alrahis, O. Sinanoglu, H. Amrouch, HDCircuit: Brain-Inspired HyperDimensional Computing for Circuit Recognition, in: 2024 Design, Automation & Test in Europe Conference &

Exhibition (DATE), IEEE, Valencia, Spain, 2024, pp. 1–2. `doi:10.23919/DATE58400.2024.10546587`.
URL `https://ieeexplore.ieee.org/document/10546587/`

[5] L. Ge, K. K. Parhi, Classification using hyperdimensional computing: A review, IEEE Circuits and Systems Magazine 20 (2) (2020) 30–47. `doi:10.1109/mcas.2020.2988388`.
URL `http://dx.doi.org/10.1109/MCAS.2020.2988388`

[6] Z. Luo, T. S. Hy, P. Tabaghi, M. Defferrard, E. Rezaei, R. M. Carey, R. Davis, R. Jain, Y. Wang, De-hnn: An effective neural model for circuit netlist representation, in: International Conference on Artificial Intelligence and Statistics, PMLR, 2024, pp. 4258–4266.

[7] J. Kim, S. Oh, S. Cho, S. Hong, Equivariant hypergraph neural networks, in: European Conference on Computer Vision, Springer, 2022, pp. 86–103.

[8] X. Jiang, Y. Zhao, Y. Lin, R. Wang, R. Huang, et al., Circuitnet 2.0: An advanced dataset for promoting machine learning innovations in realistic chip design environment, in: The Twelfth International Conference on Learning Representations, 2023.

[9] D. Skouson, A. Keller, M. Wirthlin, Netlist analysis and transformations using spydrnet, in: Proceedings of the Python in Science Conference, 2020.

[10] J. de Muijnck-Hughes, W. Vanderbauwhede, Wiring Circuits Is Easy as $\{0, 1, \omega\}$, or Is It..., in: K. Ali, G. Salvaneschi (Eds.), 37th European Conference on Object-Oriented Programming (ECOOP 2023), Vol. 263 of Leibniz International Proceedings in Informatics (LIPIcs), Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 2023, pp. 8:1–8:28. `doi:10.4230/LIPIcs.ECOOP.2023.8`.
URL `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.ECOOP.2023.8`

[11] K. Datta, I. Sengupta, H. Rahaman, A post-synthesis optimization technique for reversible circuits exploiting negative control lines, IEEE Transactions on Computers 64 (4) (2015) 1208–1214. `doi:10.1109/TC.2014.2315641`.

[12] A. Beg, A. Elchouemi, R. Beg, A collaborative platform for facilitating standard cell characterization, in: Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2013, pp. 202–206. `doi:10.1109/CSCWD.2013.6580963`.

[13] M. Hutton, J. Rose, J. Grossman, D. Corneil, Characterization and parameterized generation of synthetic combinational benchmark circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 17 (10) (1998) 985–996. `doi:10.1109/43.728919`.

[14] M. Zhang, Y. Chen, Link prediction based on graph neural networks, Advances in neural information processing systems 31 (2018).

[15] H. Sarhan, A. Arriordaz, Automated analog design constraint checking, `https://semiengineering.com/automated-analog-design-constraint-checking/` (February 2019).

[16] J. I. Hibshman, T. Weninger, Inherent limits on topology-based link prediction, Transactions on Machine Learning Research (2023). URL `https://openreview.net/forum?id=izL3B8dPx1`

[17] J. Chen, H. He, F. Wu, J. Wang, Topology-aware correlations between relations for inductive link prediction in knowledge graphs, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 35, 2021, pp. 6271–6278.

[18] eCircuit Center, SPICE Summary — ecircuitcenter.com, `https://www.ecircuitcenter.com/SPICEsummary.htm`, [Accessed 06-02-2025].

[19] DeepSeek-AI, A. Liu, B. Feng, B. Wang, B. Wang, B. Liu, C. Zhao, C. Dengr, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Xu, H. Yang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Chen, J. Yuan, J. Qiu, J. Song, K. Dong, K. Gao, K. Guan, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Pan, R. Xu, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Zheng, T. Wang,

T. Pei, T. Yuan, T. Sun, W. L. Xiao, W. Zeng, W. An, W. Liu, W. Liang, W. Gao, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Chen, X. Nie, X. Sun, X. Wang, X. Liu, X. Xie, X. Yu, X. Song, X. Zhou, X. Yang, X. Lu, X. Su, Y. Wu, Y. K. Li, Y. X. Wei, Y. X. Zhu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Zheng, Y. Zhang, Y. Xiong, Y. Zhao, Y. He, Y. Tang, Y. Piao, Y. Dong, Y. Tan, Y. Liu, Y. Wang, Y. Guo, Y. Zhu, Y. Wang, Y. Zou, Y. Zha, Y. Ma, Y. Yan, Y. You, Y. Liu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Huang, Z. Zhang, Z. Xie, Z. Hao, Z. Shao, Z. Wen, Z. Xu, Z. Zhang, Z. Li, Z. Wang, Z. Gu, Z. Li, Z. Xie, Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model (2024). `arXiv:2405.04434`.
URL `https://arxiv.org/abs/2405.04434`

[20] DeepSeek-V2.5: A New Open-Source Model Combining General and Coding Capabilities, `https://api-docs.deepseek.com/news/news0905`, [Accessed 12-02-2025].

[21] DeepSeek-AI, A. Liu, B. Feng, B. Xue, B. Wang, B. Wu, C. Lu, C. Zhao, C. Deng, C. Zhang, C. Ruan, D. Dai, D. Guo, D. Yang, D. Chen, D. Ji, E. Li, F. Lin, F. Dai, F. Luo, G. Hao, G. Chen, G. Li, H. Zhang, H. Bao, H. Xu, H. Wang, H. Zhang, H. Ding, H. Xin, H. Gao, H. Li, H. Qu, J. L. Cai, J. Liang, J. Guo, J. Ni, J. Li, J. Wang, J. Chen, J. Chen, J. Yuan, J. Qiu, J. Li, J. Song, K. Dong, K. Hu, K. Gao, K. Guan, K. Huang, K. Yu, L. Wang, L. Zhang, L. Xu, L. Xia, L. Zhao, L. Wang, L. Zhang, M. Li, M. Wang, M. Zhang, M. Zhang, M. Tang, M. Li, N. Tian, P. Huang, P. Wang, P. Zhang, Q. Wang, Q. Zhu, Q. Chen, Q. Du, R. J. Chen, R. L. Jin, R. Ge, R. Zhang, R. Pan, R. Wang, R. Xu, R. Zhang, R. Chen, S. S. Li, S. Lu, S. Zhou, S. Chen, S. Wu, S. Ye, S. Ye, S. Ma, S. Wang, S. Zhou, S. Yu, S. Zhou, S. Pan, T. Wang, T. Yun, T. Pei, T. Sun, W. L. Xiao, W. Zeng, W. Zhao, W. An, W. Liu, W. Liang, W. Gao, W. Yu, W. Zhang, X. Q. Li, X. Jin, X. Wang, X. Bi, X. Liu, X. Wang, X. Shen, X. Chen, X. Zhang, X. Chen, X. Nie, X. Sun, X. Wang, X. Cheng, X. Liu, X. Xie, X. Liu, X. Yu, X. Song, X. Shan, X. Zhou, X. Yang, X. Li, X. Su, X. Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Y. Zhang, Y. Xu, Y. Xu, Y. Huang, Y. Li, Y. Zhao, Y. Sun, Y. Li, Y. Wang, Y. Yu, Y. Zheng, Y. Zhang, Y. Shi, Y. Xiong, Y. He, Y. Tang, Y. Piao, Y. Wang, Y. Tan, Y. Ma, Y. Liu, Y. Guo, Y. Wu,

Y. Ou, Y. Zhu, Y. Wang, Y. Gong, Y. Zou, Y. He, Y. Zha, Y. Xiong, Y. Ma, Y. Yan, Y. Luo, Y. You, Y. Liu, Y. Zhou, Z. F. Wu, Z. Z. Ren, Z. Ren, Z. Sha, Z. Fu, Z. Xu, Z. Huang, Z. Zhang, Z. Xie, Z. Zhang, Z. Hao, Z. Gou, Z. Ma, Z. Yan, Z. Shao, Z. Xu, Z. Wu, Z. Zhang, Z. Li, Z. Gu, Z. Zhu, Z. Liu, Z. Li, Z. Xie, Z. Song, Z. Gao, Z. Pan, Deepseek-v3 technical report (2025). arXiv:2412.19437.
URL https://arxiv.org/abs/2412.19437

[22] DeepSeek-V3-0324 Release, https://api-docs.deepseek.com/news/news250325, [Accessed 28-04-2025].

[23] A. Yang, A. Li, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Gao, C. Huang, C. Lv, C. Zheng, D. Liu, F. Zhou, F. Huang, F. Hu, H. Ge, H. Wei, H. Lin, J. Tang, J. Yang, J. Tu, J. Zhang, J. Yang, J. Yang, J. Zhou, J. Zhou, J. Lin, K. Dang, K. Bao, K. Yang, L. Yu, L. Deng, M. Li, M. Xue, M. Li, P. Zhang, P. Wang, Q. Zhu, R. Men, R. Gao, S. Liu, S. Luo, T. Li, T. Tang, W. Yin, X. Ren, X. Wang, X. Zhang, X. Ren, Y. Fan, Y. Su, Y. Zhang, Y. Zhang, Y. Wan, Y. Liu, Z. Wang, Z. Cui, Z. Zhang, Z. Zhou, Z. Qiu, Qwen3 technical report (2025). arXiv:2505.09388.
URL https://arxiv.org/abs/2505.09388

[24] N. C. of Technology Innovation for EDA, 2024 China Postgraduate IC Innovation Competition: EDA Elite Challenge Q10 Guide, https://edaoss.icisc.cn/file/cacheFile/2024/8/7/905c4088385441fea110889b1fdeb30d.pdf (2024).

[25] F. Salvaire, Pyspice, accessed: 2025-02-12.
URL https://pyspice.fabrice-salvaire.fr/

[26] A. Said, M. Shabbir, B. Broll, W. Abbas, P. Völgyesi, X. Koutsoukos, Circuit design completion using graph neural networks, Neural Computing and Applications 35 (16) (2023) 12145–12157. doi:10.1007/s00521-023-08346-x.
URL https://link.springer.com/10.1007/s00521-023-08346-x

[27] Z. Tao, Y. Shi, Y. Huo, R. Ye, Z. Li, L. Huang, C. Wu, N. Bai, Z. Yu, T.-J. Lin, et al., Amsnet: Netlist dataset for ams circuits, in: 2024 IEEE LLM Aided Design Workshop (LAD), IEEE, 2024, pp. 1–5.

[28] 2024 China Postgraduate IC Innovation Competition·EDA Elite Challenge Contest (2024).
URL `http://edachallenge.cn`

[29] J. Bhandari, V. Bhat, Y. He, H. Rahmani, S. Garg, R. Karri, Masala-chai: A large-scale spice netlist dataset for analog circuits by harnessing ai (2025). `arXiv:2411.14299`.
URL `https://arxiv.org/abs/2411.14299`

[30] J. Gao, W. Cao, J. Yang, X. Zhang, Analoggenie: A generative engine for automatic discovery of analog circuit topologies, in: The Thirteenth International Conference on Learning Representations, 2025.
URL `https://openreview.net/forum?id=jCPak79Kev`

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio, Graph Attention Networks, International Conference on Learning Representations (2018).
URL `https://openreview.net/forum?id=rJXMpikCZ`

[32] X. Jiang, R. Zhu, P. Ji, S. Li, Co-embedding of nodes and edges with graph neural networks, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (6) (2020) 7075–7086.