IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 October 2024, accepted 7 November 2024, date of publication 13 November 2024, date of current version 22 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3496917

RESEARCH ARTICLE

Circuit2Graph: Diodes as Asymmetric Directional Nodes

YUSUKE YAMAKAJI[1,2], HAYARU SHOUNO[2], (Member, IEEE), AND KUNIHIKO FUKUSHIMA[3], (Member, IEEE)

[1]Information Technology Research and Development Center, Mitsubishi Electric Corporation, Kamakura 247-8501, Japan
[2]Graduate School of Informatics and Engineering, The University of Electro-Communications, Chofu, Tokyo 182-8585, Japan
[3]Fuzzy Logic Systems Institute, Iizuka, Fukuoka 820-0067, Japan

Corresponding authors: Yusuke Yamakaji (Yamakaji.Yusuke@dh.MitsubishiElectric.co.jp), Hayaru Shouno (shouno@uec.ac.jp), and Kunihiko Fukushima (fukushima@m.ieice.org)

ABSTRACT Circuit design requires both prototyping and simulation owing to the high degrees of freedom and mutual interference between the circuit components and wiring. In this paper, we propose a novel approach that involves transforming a diode into nodes within a graph network. This transformation considers the asymmetric directional characteristics of diodes in the forward and reverse directions. To evaluate the accuracy of our proposed method, we utilized graph classification with a graph neural network (GNN). Since a diode node holds two pieces of information, an anode and a cathode, the diode node is decomposed into two nodes, and the node attributes of the anode and cathode are assigned to each. By maintaining fixed hyperparameters and introducing a directional GNN and diode decomposition, the proposed method achieved an average accuracy of 98.80%. This accuracy surpasses the best accuracy reported in previous studies on graph classification by 0.9%. Notably, as with graph classification, changes in the handling of features obtained with GNN make it possible, for example, to optimize circuit topology and components and to predict the area required for implementation. Our method provides an environment for solving circuits on the GNN platform, enabling the prediction of values and states that could not be obtained with existing circuit simulations.

INDEX TERMS Asymmetric node, directional graph neural networks, directional node, node separation, series-parallel decomposition.

I. INTRODUCTION

Circuits serve as the foundation of electrical systems, playing a vital role in electronics. However, the process of enhancing circuits necessitates extensive simulations and relies significantly on empirical rules. Therefore, streamlining these rules or establishing an automated system that can surpass them is imperative. To address this need, the integration of circuits and neural networks emerges as a promising solution. Neural networks, inspired by brain research [1], have proven to be highly effective when combined with backpropagation [2] and large datasets. They have surpassed human performance in tasks involving datasets with coordinate systems, such as images and natural language processing [3]. Various methods based on convolutional neural networks (CNNs) [4], transformers [5], and multilayer perception (MLP) [6] have demonstrated superior recognition and generation capabilities compared with the average human in image-related tasks. In natural language processing, transformer-based [7] methods, such as GPT-3 [8] and Copilot [9] have outperformed humans in tasks related to recognition and generation.

Unlike images and natural languages, circuits lack spatial coordinates and should be treated as graph networks that do not rely on a coordinate system [10], [11] [12]. Unlike continuous values in images and natural languages, graph networks, characterized by discrete features, offer flexibility when transforming circuits into graph networks. Conventionally, circuit components and wiring are defined as nodes, with edges connecting these nodes [13], [14]. However, the

The associate editor coordinating the review of this manuscript and approving it for publication was Harikrishnan Ramiah.

distinct roles of circuit components and wiring necessitate a constraint that mandates the alternating appearances of the component and wiring nodes in these transformations. Training a GNN under these constraints presents a challenge as it involves solving a specific heterogeneous graph using a GNN. Therefore, a homogeneous approach is employed, where circuit components serve as nodes, and wirings act as edges to address this challenge.

GNNs can be applied to handle the directionality of diodes. However, one type of GNN, known as the directional GNN, only addresses directional edges and cannot be applied to diodes where the node itself has directionality. To address this issue, two potential solutions have been identified: introducing a novel directional GNN algorithm capable of handling directional nodes or modifying the structure and node attributes of the graph network, which serve as input data for the GNN, and applying them to directional GNNs. Because the ratio of diodes in a circuit is relatively small, and many circuit components and wiring lack directional information, a method based on modifying the graph structure is proposed as a viable solution. This method allows for the implementation of directional GNN algorithms without necessitating changes to the algorithms themselves, making it a seamless extension of existing examples. The simplest modification involves configuring the graph network with inward edges leading to the anode of a diode and outward edges leading to the cathode. However, this modification only accounts for forward bias from the anode to the cathode, neglecting the reverse bias of the diode.

Consequently, diodes that actively use reverse bias, such as those used for semiconductor overvoltage protection, preventing the reverse connection of DC power supplies, constant-voltage diodes utilizing the steep reverse bias characteristics of Zener diodes, and flywheel diodes for releasing accumulated energy in inductance, cannot be represented. The challenge of representing the asymmetry of a node with directed edges stems from the dual nature of information contained within a diode. A previous study [15] highlighted that semiconductors possess more than three terminals, allowing them to be decomposed into the equivalent number of nodes corresponding to the semiconductor's terminal count. By implementing the concept of transforming a node with multiple pieces of information into multiple nodes with a single piece of information, a diode node with two pieces of information (anode and cathode) is divided into two nodes. When the direction of the edges is disregarded, the two nodes can be combined in three possible configurations, as shown in Fig. 1.

The first combination involved connecting the anode and cathode nodes in series. In this configuration, signal propagation between the anode and cathode nodes is asymmetrical owing to the nonlinearity of the activation function in GNNs. Consequently, even in undirected GNNs, directional information can be retained in the hidden layers by assigning unique one-hot vectors as node attributes to the anode and cathode. The second combination involved positioning the

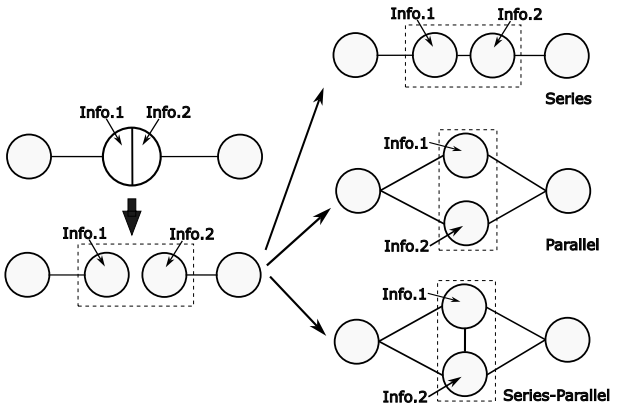


FIGURE 1. One node with two pieces of information is decomposed into two nodes with one piece of information. These nodes can be connected in three possible ways: series, parallel, and series-parallel.

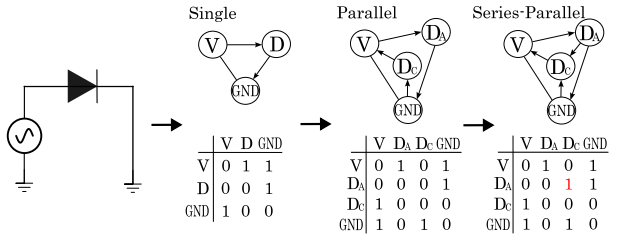


FIGURE 2. Adjacency matrix for a graph network that represents a circuit with diodes as nodes in series, parallel, or series-parallel. In this case, D_A represents the anode node, and D_C represents the cathode node.

anode and cathode nodes in parallel. In this configuration, signal propagation between the anode and the cathode is disrupted, allowing for the complete separation of the forward and reverse bias paths by assigning distinct node attributes to each node and setting the edge direction connecting the two nodes to be opposite. However, in a circuit with multiple parallel diodes, the transformation between the circuit and graph network becomes irreversible as the pairs of anode and cathode nodes cannot be uniquely identified. The third combination involves arranging the anode and cathode nodes in a series-parallel configuration, connecting the two nodes in parallel, and establishing a direct link between them. This configuration comprises three additional degrees of freedom in the direction of the edges between the two nodes (from the anode to cathode, cathode to anode, and in both directions).

Graph networks can be represented by adjacency matrices, which have the same number of rows and columns as the number of nodes, where elements with edges between nodes are represented as 1, and elements with no edges are represented as 0. In particular, directed graphs with one or more directional edges within a graph can be represented by setting the rows as sources and the columns as sinks, and non-directional edges can be defined as bidirectional edges. Fig. 2 shows an example of a simple circuit with a diode expressed as a series, parallel, and series-parallel node using adjacency matrices. As shown in the figure, an adjacency matrix for a graph with directional edges is an asymmetric matrix. The "Single" is sufficient to express an ideal diode.

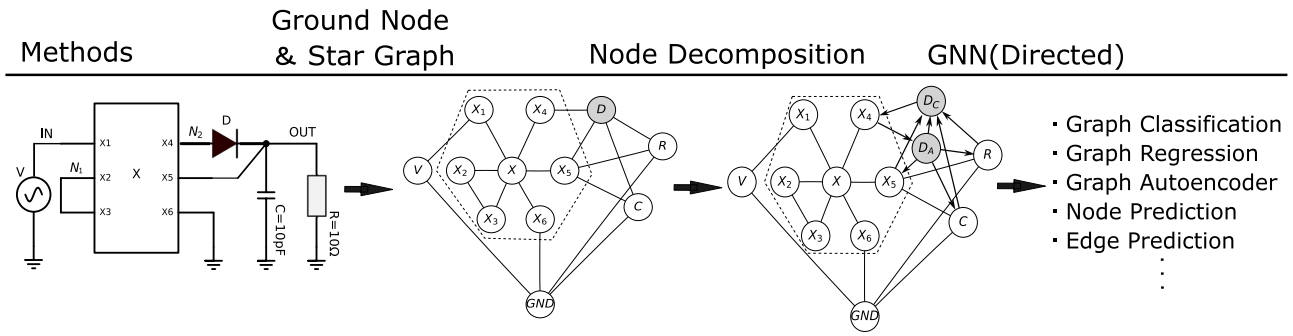


FIGURE 3. Transformation from a circuit to a graph network with diodes: the first transformation incorporates a ground node to the circuit and decomposes the semiconductor node into a star graph using a virtual node. In the second transformation, the diode node is decomposed into two nodes and connected in a series-parallel manner.

However, the ''Parallel'' and ''Series-Parallel'' are necessary for accurately modeling the operation of an actual diode that takes into account the reverse bias characteristics.

We hypothesize that minimizing information loss during the transformation from circuits to graph networks with these graph network modifications will enhance the accuracy of inference in GNNs. By retaining essential information for feature extraction within the graph network, we anticipate an improvement in inference accuracy. To test our hypothesis, we conducted a comparative analysis between diode representation and the graph classification accuracy of GNNs. We will explain the highlights of our study utilizing Fig. 3. In the figure, we incorporate a ground node into the graph network, which uses circuit components as nodes, and decompose the semiconductor into a star graph to reduce self-loops and multiple edges and to preserve the terminal numbers of the semiconductor within the graph network. We then transform the diode nodes into a series-parallel format and assign directed edges. In particular, we define directed edges from the anode to the cathode of the diode to make the diode an asymmetric structure.

The contributions of this study are as follows:

1) We propose a method for decomposing a node containing two pieces of information into two nodes containing a single piece of information each, enabling the use of directional GNNs to handle directional nodes that can only treat directional edges.
2) Under the constraint of reversibility between circuits and graph networks in the homogeneous graph, unique one-hot vectors were assigned to the node attributes of the two nodes.
3) By applying a directed GNN to the aforementioned graph network, we successfully extracted high-quality circuit feature values, leading to enhanced inference accuracy.

A. PREVIOUS STUDIES

The social demand for automated circuit designs has increased, with GNNs emerging as a promising solution for semiconductor applications [16], [17] [18], [19] [20]. However, challenges, such as stringent design rules, rising design costs owing to the miniaturization and adoption of low-voltage transistors have limited the widespread adoption of GNNs in the industry [11], [21]. For example, commercial electronic design automation tools provided by Synopsys, Inc., Cadence Design Systems, Inc., Siemens, Inc., and Zuken, Inc. have been employed for simulations, logic designs, and analog designs. However, these tools, which rely on design rules and simulations, require technical expertise and incur simulation costs during parameter searching. Therefore, leveraging previous designs and simulation data is crucial to alleviate these challenges. Examples include the integration of logic synthesis [17], [22], [23], reinforcement learning for field-programmable gate-array performance [23], and transistor placement [24]. Notably, these examples do not necessarily consider circuit and graph network reversibility, as the output typically comprises images of the component layout [24] and numerical simulation values [13].

In addition, previous studies have focused on the internal design of semiconductors, which may not be directly applicable to general-purpose electrical circuits, such as printed circuit boards, that include analog circuits. In particular, several studies [25], [26] assumed that the internal circuit and characteristics of semiconductors are known during the design phase. However, the internal circuits of semiconductors are not disclosed to users, and circuit simulations utilizing such semiconductors are challenging unless the semiconductor manufacturer publishes the models. However, treating circuits as graph networks allows for the handling of circuit components with unknown characteristics.

The transformation of circuit components into graph networks without abstraction can be realized using three methods. In [15], [25], and [27], circuit components were represented as nodes and wiring as edges. In [13] and [28], both circuit components and wiring were represented as nodes, whereas [29] and [30], represented the wiring as a node and circuit components as edges. Alternatively, the circuit components can be abstracted into a graph network

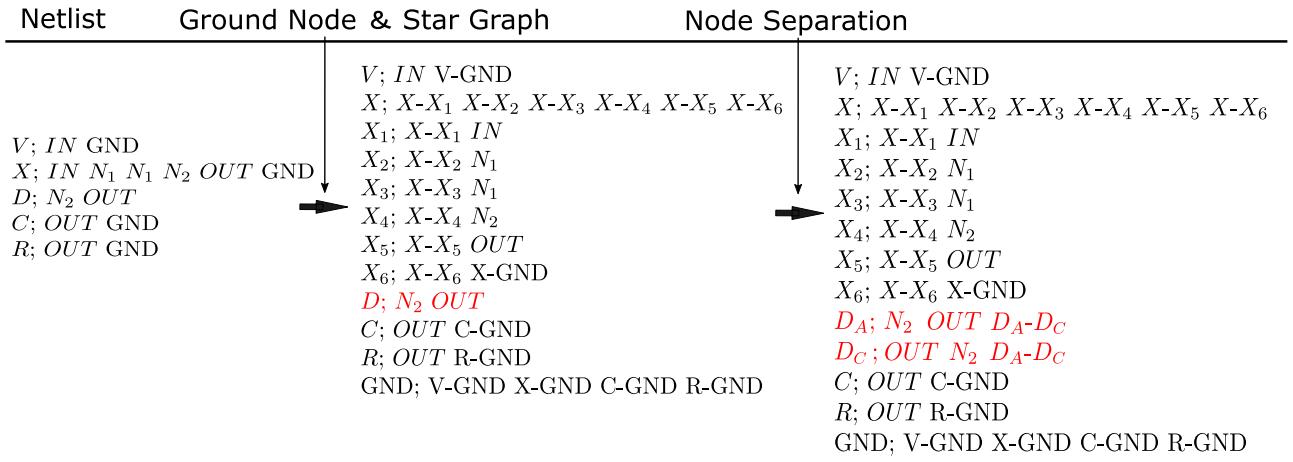


FIGURE 4. Circuit to netlist transformation: the first transformation incorporates a ground node to the netlist and transforms the semiconductor node into a star graph [15]; the second transformation decomposes a diode into two circuit components, that is, an anode node D_A and a cathode node D_C.

by replacing multiple components with a single node. This replacement allows for the inclusion of components with unknown characteristics into the graph network [31], [32]. However, neither method could transform a graph network without losing information regarding a diode with node directionality. Therefore, this study demonstrates the effect of directed nodes within a homogeneous and simple graph network [15]. In this network, circuit components and grounds serve as nodes, wiring (excluding grounds) as edges, and semiconductors as star graphs.

B. GNN ALGORITHM

Various undirected GNNs have been developed, such as GCN [33], GAT [34], and GraphSage [35]. Among these, we evaluated the GCN, which is a fundamental GNN algorithm, and GraphSage, which balances the training speed and inference accuracy through node sampling in hidden layers. Conversely, fewer methods exist for directed GNNs than those for undirected GNNs. DiGCN [36], for example, instead of normalizing by the degree of the adjacency matrix as in GCN, it is normalized by the diagonal elements of the adjacency matrix, preserving the asymmetry of the adjacency matrix and directed edges. Magnet [37] maintains directed edges by assigning edge weights and directions based on amplitude and phase using the magnetic Laplacian, which is a complex Hermitian matrix with complex numbers. In a Dir-GNN, the ratio of the inward and outward edges is determined as a hyperparameter during training to enhance inference accuracy. However, to highlight the difference in inference accuracy under a fixed ratio and to minimize the convergence time during training, we manually set the ratio as a hyperparameter. Furthermore, we utilize undirected and directed GNN algorithms on circuits, employing one-hot embedding vectors [15] that enable circuit constants to be directly assigned to node attributes. In Dir-GNN [38], feature values are extracted from the graph network by applying different undirected GNNs to inward and outward edges, merging the features of the two GNNs to form a single feature. By dividing the inward and outward directions, the undirected graph can be applied to each direction, which is a natural extension of undirected GNNs. In addition, since the algorithm of the undirected GNN can be directly integrated, we assess "Dir-SAGE," which incorporates GraphSage, that achieved the optimal results in undirected GNNs. Furthermore, Dir-GNN improves inference accuracy by determining the ratio of inward and outward edges as a hyperparameter during training. However, to highlight the difference in inference accuracy under a fixed ratio and to minimize the convergence time during training, we manually set the ratio as a hyperparameter. Furthermore, we utilize undirected and directed GNN algorithms on circuits, employing one-hot embedding vectors [15] that enable circuit constants to be directly assigned to node attributes.

II. NODE TRANSFORMATION OF DIODE

Circuits can be transformed into graph networks using two methods. One method involves modifying the netlist, which is a text data format that describes the circuit components and their connections. The other method is to modify the graph network after transforming it from the netlist. Both approaches yield similar results; however, for clarity, an example of modifying the former is shown in Fig. 4. There are several netlist formats, including Allegro, Telesis, Scicards, and LTspice; however, because transformation between netlists is possible without loss of details, a standard LTspice [39] format was selected. Because the edge direction is determined by the order of the wirings listed in a netlist, it is transformed according to the definition of the description order during the transformation to the adjacency matrix, representing the node connection of the graph network. In Fig. 4, for the anode node D_A, the wirings are described in the order N_2, OUT, D_A-D_C, where N_2 represents the

input wiring, OUT represent the output wiring, and D_A-D_C represents the wiring connected to D_C node. For the cathode node D_C, based on the same definition as D_A, OUT represents the input wiring, N_2 represents the output wiring, and D_A-D_C represents the wiring connected to the D_A node. However, a new challenge emerges when a diode node is divided into two, as the two nodes can be flexibly connected in series, parallel, or a combination of both.

A. SINGLE NODE

For increased comprehension of the following three methods, we consider a scenario in which the diode node remains undivided. The simplest extension for representing directed nodes with directed edges is to represent a diode with a single node and connect the directed edges to that node. The graph network with the circuit shown in Fig. 3 as a single node and directed edges is shown in Fig. 5, according to which the graph of the diode is formed by edges from X_4 to D and from D to X_5, C, and R nodes. Any wiring that is not directional, other than the wiring connected to the diode, is considered bidirectional. This definition allows signal transmission between nodes, even in a directional GNN, ensuring that circuits and graph networks share the same characteristics.

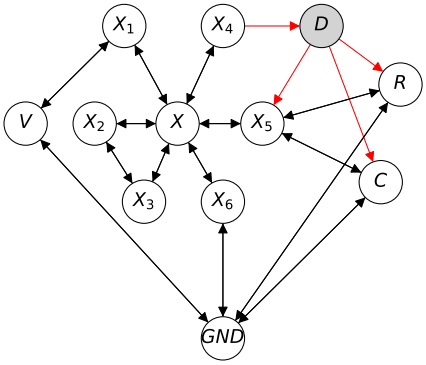


FIGURE 5. Graph network with a single diode and directed edges.

Signal transmission in the hidden layer of the GNN around the diode can be expressed using eq. (1), where σ represents an activation function (a nonlinear function), $\mathbf{W}$ represents the weight matrix that is optimized during GNN training, and $\mathbf{x}$ represents the vector of node attributes of the diode. As shown in eq. (1), the directional edge allows for signal propagation in the forward direction but not in the reverse direction.

$$\mathbf{u}(\mathbf{x}) = \begin{cases} \sigma(\mathbf{W} \cdot \mathbf{x}) & \text{if forward bias} \\ 0 & \text{if reverse bias} \end{cases} \tag{1}$$

B. SERIES NODE

When a diode is divided into two nodes, the simplest and most intuitive connection is to link the two nodes in series. This transformation enables the division of the characteristics between nodes D_A and D_C. GNNs with a directed edge from D_A to D_C cannot train the reverse bias characteristic because the signal does not propagate from D_C to D_A. Therefore, to accommodate both forward and reverse directions, the edge between D_A and D_C should be designated as bidirectional. Consequently, all edges within a graph network are bidirectional, allowing them to be trained by undirected GNNs. However, the paths in the forward and reverse directions are identical, presenting a challenge in sharing the weight matrix of the GNN in both directions. By dividing the diode node and assigning different one-hot vectors to each node attribute, the forward and reverse characteristics can be asymmetrically owing to the activation function of GNNs. However, to achieve the same level of inference accuracy without information loss, the amount of data required for training and computational costs must be increased.

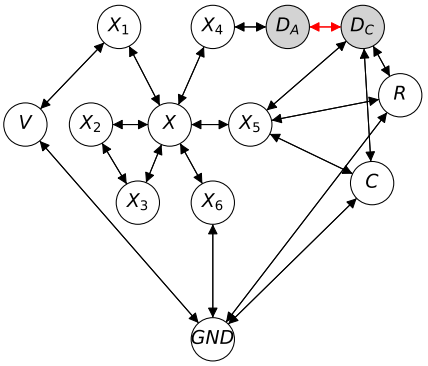


FIGURE 6. Graph network with diodes as series nodes and an undirected graph.

The diode can be split into two and defined as a series node, with the hidden layer of the GNN represented using the eq. (2). In this equation, the forward and reverse directions share a weight matrix $\mathbf{W}$, and only the sequence in which the vectors of the node attributes $\mathbf{x_A}$ of D_A and $\mathbf{x_C}$ of D_C are applied to $\mathbf{W}$ varies. Although the activation function σ results in different signal propagation in the forward and reverse directions, it cannot sufficiently address the asymmetrical relationship between the forward and reverse threshold voltages in actual diodes. Therefore, while this approach maintains the relationship between the two nodes, it cannot train independent characteristics owing to strong dependencies in the forward and reverse directions.

$$\mathbf{u}(\mathbf{x}) = \begin{cases} \mathbf{x_C} + \sigma(\mathbf{W} \cdot \mathbf{x_A}) & \text{if forward bias} \\ \mathbf{x_A} + \sigma(\mathbf{W} \cdot \mathbf{x_C}) & \text{if reverse bias} \end{cases} \tag{2}$$

The results of the node attributes for each node in Fig. 6, expressed as one-hot embedding vectors, are listed in Table 1, according to which the node attributes of the terminal nodes of a semiconductor and a virtual node are assigned as a one-hot vector, and circuit constants of "C" and "R" are assigned as a one-hot embedded vector, with a "1" element in the vector representing a real number. The logarithms of the circuit constants are obtained and normalized by the

minimum and maximum values of "C" and "R" in the dataset. This process ensures that each circuit constant fits within the range of -1 to +1. Furthermore, D_A and D_C are assigned distinct one-hot vectors, such that their node attributes are orthogonal in the Adamar product [40]. The orthogonality of the one-hot and the one-hot embedding vectors enables the propagation of the node attributes of D_A to D_C in the hidden layer of the GNN to persist even when more layers of the GNN are added. In addition, because the dataset utilized for the evaluation did not include more than nine semiconductors in a single circuit, a one-hot embedded vector was employed to assign the value $y = (1.0 - 0.1 \times n)$ to the n-th semiconductor.

TABLE 1. Node attributes based on one-hot embedding vectors.

X	1	0	0	0	0	0	0
X:1	1	0	0	0	0	0	0
X:2	1	0	0	0	0	0	0
X:3	1	0	0	0	0	0	0
X:4	1	0	0	0	0	0	0
X:5	1	0	0	0	0	0	0
X:6	1	0	0	0	0	0	0
V	0	1	0	0	0	0	0
D_A	0	0	**1**	0	0	0	0
D_C	0	0	0	**1**	0	0	0
C	0	0	0	0	-0.85	0	0
R	0	0	0	0	0	0.08	0
GND	0	0	0	0	0	0	1

C. PARALLEL NODE

One challenge faced is the inability to separate the forward and reverse paths cannot in series. This challenge can be resolved by placing two nodes in parallel and assigning edges in different right and left directions to each parallel node. A graph network with this configuration is shown in Fig. 7 where D_A receives signals from the X_4 node and outputs them to the X_5, C, and R nodes. Conversely, D_C receives signals from the X_5, C, and R nodes and outputs them to the X_4 node. No direct edge exist between D_C and D_A; they are connected through the adjacent nodes: X_4, X_5, C, and R. Although no direct edges exist between D_C and D_A, allowing for complete separation of the forward and reverse paths, D_C and D_A become independent circuit components. Therefore, a circuit with two parallel-connected diodes loses the reversibility between the graph network and a circuit.

By partitioning the diode into two segments and them as parallel nodes, and describing the GNN's hidden layer in relation to these two nodes, we can express the relationship using the eq. (3) where the forward and reverse directions exhibit different weight matrices $\mathbf{W_1}$ and $\mathbf{W_2}$, respectively. The vector of node attributes $\mathbf{x_A}$ of D_A and $\mathbf{W_1}$ as well as the vector of node attributes $\mathbf{x_C}$ of D_C and $\mathbf{W_2}$, may exhibit entirely different characteristics.

$$\mathbf{u(x)} = \begin{cases} \sigma\left(\mathbf{W_1} \cdot \mathbf{x_A}\right) & \text{if forward bias} \\ \sigma\left(\mathbf{W_2} \cdot \mathbf{x_C}\right) & \text{if reverse bias} \end{cases} \tag{3}$$

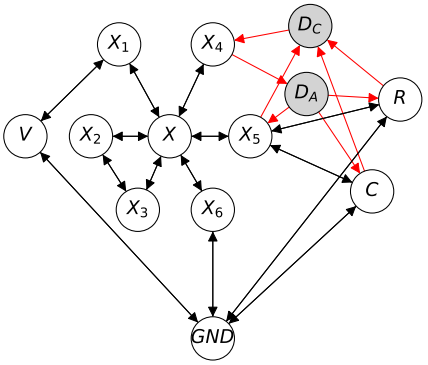


FIGURE 7. Graph network represented by direct edges with diodes as parallel nodes.

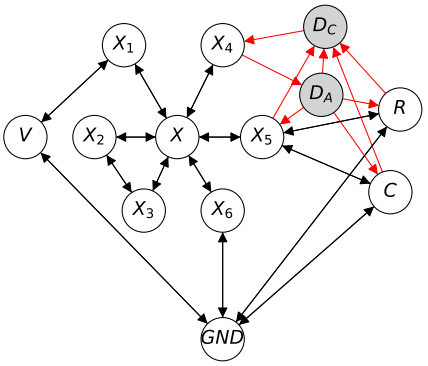


FIGURE 8. Graph network representing a diode as a series-parallel connection from anode to cathode nodes.

D. SERIES-PARALLEL NODE

To address the challenges posed by series and parallel nodes, we propose series-parallel nodes. These nodes offer three potential methods for defining the edges between them within a series-parallel node structure.

1) an asymmetric structure with directed edges from D_A to D_C.
2) an asymmetric structure with directed edges from D_C to D_A.
3) a symmetric structure of bidirectional directed edges between D_A and D_C.

A graph network with edges extending from D_A to D_C is shown in Fig. 8. In 1) and 2), distinct paths can be asymmetrically defined in the forward and reverse directions, enabling the determination of the diode direction from the graph network structure without relying on the node attributes of D_A and D_C. This structure enables reversibility from the graph network to a circuit, even in a circuit with multiple parallel-connected diodes, utilizing a graph structure alone.

By dividing the diode into two parts, defining them as series-parallel nodes, and elucidating the GNN's hidden layer in relation to these two nodes, the relationship can be expressed using eq. (4). In this equation, the forward and reverse directions are distinguished by weight matrices $\mathbf{W_1}$ and $\mathbf{W_2}$, respectively, with a weight matrix $\mathbf{W_3}$ between D_A

TABLE 2. Dataset: Average node and edge values for seven different circuits as well as the number of circuits containing a diode when a single node represents the diode.

	ADC	Comp.	Filter Products	Opamps	Power Products	Ref.	Switches
Number of Circuits	36	41	25	681	2240	66	119
Circuits with diode	0	1	0	36	855	1	0
Avg. Diodes	0.00	6.00	0.00	2.86	1.97	2.00	0.00
Avg. Nodes(Single)	27.94	16.76	27.32	18.15	28.90	11.08	16.76
Avg. Nodes (Series&Parallel& Series-Parallel)	27.94	16.90	27.32	18.30	29.63	11.11	16.76
Avg. Edges(Single)	47.22	35.80	52.68	39.49	65.90	21.95	33.60
Avg. Edges(Series)	47.22	36.63	52.68	40.04	69.62	22.04	33.60
Avg. Edges(Parallel)	47.22	36.56	52.68	40.07	69.63	22.08	33.60
Avg. Edges (Series-Parallel)	47.22	36.70	52.68	40.18	70.11	22.14	33.60

and D_C. In the forward direction, a signal comprising D_A and $\mathbf{W_1}$ propagates to adjacent nodes. In contrast, in the reverse direction, the signal $\mathbf{L_1} := \sigma(\mathbf{W_3} \cdot \mathbf{x_A})$ from D_A to D_C obtained in the first hidden layer and $\sigma(\mathbf{W_2} \cdot \mathbf{L_1})$ obtained in the second hidden layer propagate to the adjacent cathode nodes.

$$\mathbf{u(x)} = \begin{cases} \sigma(\mathbf{W_1} \cdot \mathbf{x_A}) & \text{if forward bias} \\ \sigma(\mathbf{W_2} \cdot (\mathbf{x_C} + \sigma(\mathbf{W_3} \cdot \mathbf{x_A}))) & \text{if reverse bias} \end{cases} \tag{4}$$

Similarly, a graph network with the directions of D_A and D_C is represented using eq. (5). Furthermore, if the edge between D_A and D_C is bidirectional, the equation becomes a combination of the forward bias in eq. (5) and the reverse bias in eq. (4).

$$\mathbf{u(x)} = \begin{cases} \sigma(\mathbf{W_1} \cdot (\mathbf{x_A} + \sigma(\mathbf{W_3} \cdot \mathbf{x_C})) & \text{if forward bias} \\ \sigma(\mathbf{W_2} \cdot \mathbf{x_C}) & \text{if reverse bias} \end{cases} \tag{5}$$

III. EXPERIMENTS

A. DATASET

The netlist, which contains all information required to construct the circuit, is not publicly available. To date, research has been conducted in collaboration with semiconductor manufacturers. Only 60 datasets have been made publicly available [41], comprising ladder circuits and operational amplifiers. However, these datasets are insufficient for training GNNs as they tend to converge to a local minimum, leading to a high variability in results. Furthermore, because these datasets are published in a graph network format, circuit-to-graph network transformation becomes impossible. Therefore, sample circuits [42] provided by LTspice [39], a free commercially available circuit simulator from Analog Devices, Inc. were utilized. The dataset was classified into seven types [15] based on the semiconductors used in the circuits. The seven sample circuits comprised 3,308 different semiconductors with varying model numbers, of which 2,315 (70%) were used as training data and the remaining 993 as test data. In a typical circuit, the position of the "1" in the one-hot vector should vary depending on the semiconductor type, such as the power supply, operational amplifier, or switch. However, in the GNN classification of the sample circuit, the correct label is the same as that of the semiconductor type. By assigning the semiconductor type to the node attribute, the GNN no longer needs to extract the circuit features. Therefore, as shown in Table 1, a one-hot vector is assigned to the node attributes of the semiconductor terminal nodes and a virtual node, indicating only "semiconductor" rather than specifying the semiconductor type. Notably, LTspice provides 14 types of circuits for semiconductors; however, six of them were excluded because of an insufficient number of sample circuits (approximately 10), which were not suitable for training and inference. Additionally, while "SpecialFunctions" contains 193 circuits, both "SpecialFunctions" and "PowerProducts" contain circuits with the same structure, differing only in the model number of the semiconductor. Therefore, without additional information provided for the semiconductor node, it becomes impossible to differentiate between each circuit using the one-hot vector, leading to the exclusion of "SpecialFunctions."

The type of LTspice utilized as datasets, the number of files and diodes in each type, and the average number of nodes and edges when representing the circuits as a graph network are listed in Table 2. "PowerProducts" contains most of the diodes, with 1,680 diodes implemented in 855 of the 2,240 circuits. These findings suggest that the diodes are contained in approximately one-third of the circuits, with an average of two diodes implemented in each circuit. Furthermore, the "PowerProducts" circuit, which contains a higher number of diodes, has an average diode count of 1.97 out of 28.90 components, equating to 6.82% in a circuit. The relatively low ratio of diodes in the circuit mitigates the drawbacks of an increase in the number of nodes and edges owing to diode decomposition, while underscoring the advantage of managing directed nodes with existing directed GNNs.

Fig. 9 shows the top ten types of nodes adjacent to the anode node and cathode nodes. The sixth result shows that

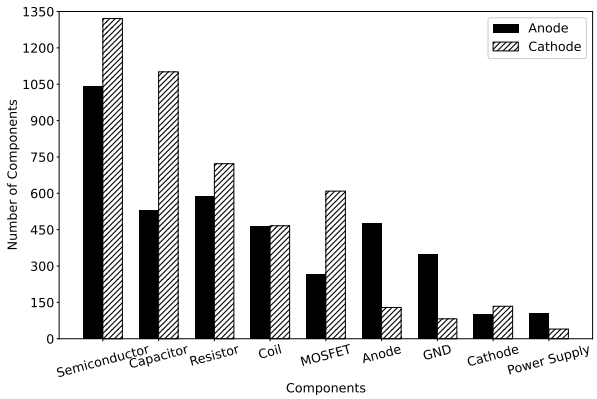


FIGURE 9. The type of node that is adjacent to the anode node and the cathode node.

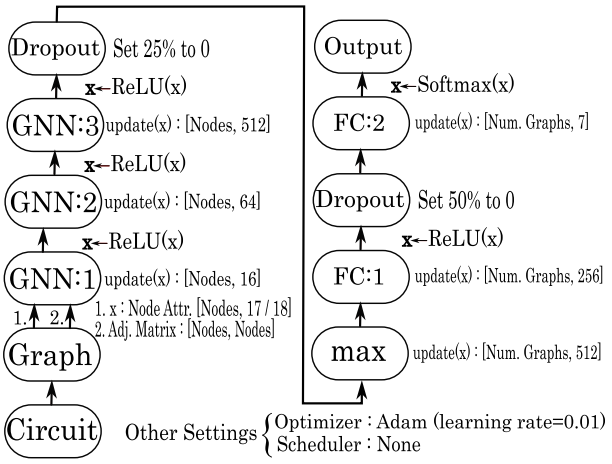


FIGURE 10. GNN model for graph classification.

the circuit contains a series or parallel connection between the anode node and the anode node of another diode. Similarly, the eighth result shows that the cathode node is connected to the cathode node of another component. This result shows that by dividing the diode into series-parallel nodes, it is possible to transform the graph back into a circuit for parallel connections.

B. HYPERPARAMETER OF GNN

Owing to the statistical nature of the GNNs, inference results exhibit variability. The accuracy of GNN inference is contingent upon the distribution of training and test data, GNN structure, GNN hyperparameters, and random initial values of the weight matrix. To reduce the variability in inference results caused by differences in GNN hyperparameters, we kept the hyperparameters constant when dividing the diode, except for the types of node attributes that increase when the diode node becomes the anode node and the cathode node. In the LTspice dataset, the node attribute when the diode is not divided is 17, and the node attribute when the diode is divided into the anode node and the cathode node is 18. Fig. 10 shows the GNN model and the hyperparameters used for graph classification. The input data is the node attributes of each node and the adjacency matrix. In GNN, the node attributes are updated, but the adjacency matrix is not updated, so only the number of node attributes varies in each layer of GNN.

Additionally, we generated ten datasets with different combinations of training and test data to assess the average inference accuracy of the test data. Our goal is not to achieve the highest inference accuracy but rather to compare the difference in inference accuracy with and without diode directionality. Therefore, we intentionally refrained from hyperparameter searching and set the same conditions as those utilized in a previous study [15]. We extracted maximum feature values from the hidden layers of a three-layer GNN (with 16, 64, and 512 channels). These maximum feature values were input into two fully connected layers (with 512 and 256 channels) and aggregated into seven classes, similar to the output class. Since the mini-batch is not used, the number of graphs handled at once in training is 2,315. To prevent overfitting, a 25% dropout function was implemented in the second layer of the GNN and a 50% dropout function in the first layer of the fully connected layer. During training, the output was classified into seven classes using a softmax function for the output layer values of the fully connected layer. Both undirected and directed GNNs were utilized with the Tanh function, which has an output range of -1 to $+1$, as the activation function and Adam [43] as the optimization function. We did not employ hidden layer normalization, such as batch normalization [44] and layer normalization [45], which contribute to faster convergence owing to the challenges involved in applying them to Magnet [37], which involves complex numbers. Because the number of epochs until convergence had to be increased without batch normalization, the epochs were set to 10,000, and test data were inferred using the weight matrix from the epoch with the highest inference accuracy for training data.

The training was performed on a computer equipped with a CPU: Intel Xeon Gold 5115 and a single GPU: NVIDIA Quadro RTX 8000. The time required for training and inference was approximately 25 min per 10,000 epochs of training. The maximum GPU memory utilization for training without a mini-batch was 12GB, which is comparable to the memory capacity of a general-purpose GPU. Therefore, the training was conducted without mini-batches. However, when utilizing mini-batches, the inference accuracy decreased owing to the small amount of training data, leading to convergence to a local minimum.

C. EXPERIMENTAL RESULTS

The inference accuracy is achieved by transforming a diode into single, series, parallel, and series-parallel nodes, utilizing undirected GNN algorithms of GCN [33] and GraphSage [35], as well as directed GNN algorithms of Magnet [37], DiGCN [36], and Dir-SAGE [38] is shown in Table 3. The

TABLE 3. Experimental results: Inference accuracy of undirected and directed GNNs for transforming diodes into graph networks (average and variance of inference results utilizing ten different combinations of training and test data).

Type	GNN Algorithm	Single	Series	Parallel	Series-Parallel ($D_A \rightarrow D_C$)	Series-Parallel ($D_C \rightarrow D_A$)	Series-Parallel ($D_A \leftrightarrow D_C$)
Undirected (Baseline)	GCN [33]	97.39±0.58	97.54±0.55	97.48±0.57	97.46±0.68	97.48±0.54	97.50±0.69
	GraphSage [35]	97.89±0.87	97.95±0.62	97.98±0.68	97.99±0.56	97.95±0.76	97.97±0.64
Directed	Magnet [37]	98.09±0.68	98.33±1.36	98.12±0.69	98.28±1.21	97.93±0.71	98.10±0.47
	DiGCN [36]	97.96±1.32	97.80±0.75	97.90±0.76	98.05±1.23	97.55±1.10	97.89±1.08
	Dir-SAGE [38] ($\alpha = 0.9$)	98.39±1.51	98.29±1.50	98.14±0.71	**98.80±0.92**	97.84±0.63	98.25±1.34

TABLE 4. Experimental results for the hyperparameter α of Dir-SAGE, which applies GraphSage [35] to Dir-GNN [38].

α	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Test Accuracy[%]	98.53±1.06	98.70±0.99	98.62±0.99	98.61±0.98	98.45±0.73	98.50±1.10	98.52±0.97	98.60±0.78	98.80±0.92

TABLE 5. Experimental results: Average error rates for undirected and directed GNNs on the test dataset of circuits containing diodes.

Type	GNN Algorithm	Single	Series	Parallel	Series-Parallel ($D_A \rightarrow D_C$)	Series-Parallel ($D_C \rightarrow D_A$)	Series-Parallel ($D_A \leftrightarrow D_C$)
Undirected (Baseline)	GCN [33]	0.57 %	0.45 %	0.34 %	0.57 %	0.57 %	0.45 %
	GraphSage [35]	0.45 %	0.23 %	0.23 %	0.68 %	0.68 %	0.68 %
Directed	Magnet [37]	0.34 %	0.23 %	0.23 %	0.11 %	0.34 %	0.23 %
	DiGCN [36]	0.34 %	0.45 %	0.57 %	**0.00** %	0.23 %	0.34 %
	Dir-SAGE [38] ($\alpha = 0.9$)	0.11 %	0.34 %	0.11 %	**0.00** %	0.11 %	**0.00** %

baseline accuracy was achieved by combining single-node diode representation and GraphSage, resulting in a value of 97.89%. When the number of channels in the hidden layer, activation function, and optimization function were maintained constant, and only the GNN algorithm was changed, training with directed GNN for series-parallel nodes ($D_A \rightarrow D_C$) enhanced the inference accuracy, regardless of the algorithms. Dir-SAGE [38] achieved an average inference accuracy of 98.80%. The "rightward" wiring around a diode in the circuit defines the "outward" edges, and the "leftward" wiring defines the "inward" edges. By setting α to 0.9, which signifies the weight ratio of features obtained from the inward and outward directions, we combined them at a ratio of 0.1 to 0.9, respectively. By adjusting the weight ratio, the accuracy was 0.9% higher than the maximum accuracy of the undirected GNN. The average inference accuracy and variation results, when the value of α was varied in Dir-SAGE, are listed in Table 4. The maximum accuracy was achieved at α = 0.9, and the inference accuracy was low at approximately α = 0.5, where the weights in the forward and reverse directions were equal. Moreover, when the direction of the edge was reversed, $D_C \rightarrow D_A$, the inference accuracy decreased by 1.0%. In addition, when the direction of the edges between the nodes was bidirectional, $D_A \leftrightarrow D_C$, the inference accuracy was 98.06%, a decrease of 0.7% from the highest accuracy. These findings suggest that, similar to the circuit characteristics, the forward bias is significant in diodes, and the accuracy of inferences decreases when the forward bias characteristic is combined with the reverse bias characteristic. Table 5 presents the average error rates for the undirected and directed GNNs for the test datasets of files containing diodes for ten different combinations of training and test data. For the circuit containing the diode, 0% error rate is observed for $D_A \leftrightarrow D_C$ and $D_A \rightarrow D_C$ respectively. On the other hand, the inference accuracy for $D_A \leftrightarrow D_C$ is 98.25%, which is about 0.6% lower than the 98.80% for $D_A \rightarrow D_C$. This difference shows that, in $D_A \leftrightarrow D_C$, the GNN is required to train with a bias towards diodes, whereas in $D_A \rightarrow D_C$, the structure of the diodes is matched to the characteristics of the circuit, so that the circuit with diodes can be treated in the same way as the circuit without diodes.

D. DISCUSSION

Magnet and DiGCN, designed with the assumption that most edges in the graph network are directed, exhibited lower accuracy compared with Dir-SAGE. Dir-SAGE combines GraphSage features with both forward and reverse characteristics, making it a suitable GNN algorithm for circuit data in which most edges are undirected. However, the result obtained with α = 0.5 differed from that of GraphSage because Dir-GNN utilized the degree of inward and outward adjacency matrices for normalization. For undirected GNNs, dividing nodes into a series enhanced the inference accuracy by 0.2%, demonstrating that dividing the diode into anode and cathode nodes can prevent information degradation even in a circuit with undirected GNNs. This result can be applied not only to diodes but also to directional nodes, such as DC power supplies and aluminum electrolytic

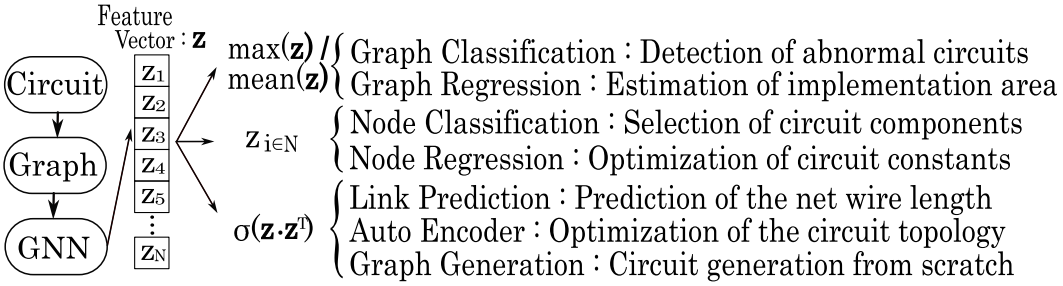


FIGURE 11. Example of applications with GNN: By extracting feature vectors z with GNN and utilizing z, the desired output can be obtained regardless of whether it is supervised or unsupervised training.

TABLE 6. Advantages and disadvantages of the method of transforming diodes to graph networks.

	Mutual Transformation	GNN Accuracy (Directed GNN)	GNN Accuracy (Undirected GNN)
Single	Good	Fair	Poor
Series	Good	Fair	Good
Parallel	Poor	Poor	Fair
Series-Parallel ($D_A \rightarrow D_C$)	Good	Good	Fair
Series-Parallel ($D_C \rightarrow D_A$)	Good	Poor	Fair
Series-Parallel ($D_A \leftrightarrow D_C$)	Poor	Fair	Fair

capacitors. However, as these circuit components do not need to include the reverse direction, they can be trained by connecting the decomposed nodes in series, assigning different node attributes to each node, and implementing an undirected or directed GNN. In cases where a diode is integrated into a semiconductor terminal as indicated by the block diagram, transforming the semiconductor into a star graph and representing the corresponding terminal as a diode node can mitigate information degradation. The advantages and disadvantages of each diode representation are listed in Table 6. For applications that do not require reverse-bias characteristics, the combination of a series node and directed GNNs provides superior results in terms of simplicity and mutual transformations between the circuit and graph network.

In addition, conventional circuit analysis and simulation techniques focus on extracting voltage, current, and impedance. However, as shown in Fig. 11, by extracting features using GNNs and defining a neural network model and evaluation function that match the training dataset, it can be applied to tasks other than graph classification. The following is an example:

1) Graph classification: Detection of abnormal circuits, prediction of the number of substrate layers, selection of manufacturing processes and materials, classification of circuits divided into subgraphs, prediction of failure modes, detection of patent violations.
2) Graph regression: Implementation area, manufacturing cost, design/manufacturing/testing period, yield rate, durability, mean time between failure, design for manufacturability, electromagnetic noise level tolerance, power supply/signal quality, net wiring length.
3) Node classification: Optimization of part numbers, extraction of parts that may fail before the design life, impact of parts on power supply quality, extraction of parts with a low tolerance to electromagnetic noise, extraction of parts that require thermal management.
4) Node Regression: Optimization of circuit constants, component failure probability, frequency/power of electromagnetic noise, frequency/power/terminals that are prone to malfunction by electromagnetic noise, rate of change by aging deterioration, parasitic capacitance between terminals, leakage current.
5) Link prediction: Bonding wire/wiring on the board/cable breakage probability, interference path prediction by parasitic capacitance/mutual inductance, wiring width optimization, prediction of residual resistance/inductance, prediction of radiated noise.
6) Graph autoencoder: Extraction of circuits with similar topologies, circuit topology optimization, block diagram construction for system design with clustering, simplification of large-scale circuits.
7) Graph generation: Generation of completely new circuits, generation of large-scale circuits by scaling down small-scale circuits, power-saving/fault-tolerance/low-cost/low-noise/noise-resistance optimization.

Solving the challenges of circuit design using and combining these applications is a future challenge.

IV. CONCLUSION

This study proposed the transformation of a diode into a graph network, allowing for reversible transformations between a circuit and a graph network. Diodes are directed asymmetric nodes; therefore, directed GNNs, which only handle directed edges, cannot be applied, so the diode was split into two nodes. These nodes were then parallel-connected with the component node adjacent to the diode, with directed edges in different directions connected to each parallel node. Furthermore, by connecting the directed edge from the anode node to the cathode node, the diode was transformed into a serial-parallel graph. This transformation enabled circuits containing diodes to be transformed into graph networks

with no information loss. Our method achieved an impressive accuracy of 98.80%, surpassing the highest accuracy reported in previous studies by 0.9% under the same hyperparameters. In addition to improving the accuracy of graph classification of circuits, the result shows that transforming circuits into graph networks is a framework for using GNNs to extract feature values that are essential for circuit analysis and design.

REFERENCES

[1] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biol. Cybern.*, vol. 36, no. 4, pp. 193–202, Apr. 1980.

[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[5] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16×16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Represent.*, 2020. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy

[6] I. O. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, "MLP-mixer: An all-MLP architecture for vision," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34. Red Hook, NY, USA: Curran Associates, 2021, pp. 24261–24272.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30. Red Hook, NY, USA: Curran Associates, 2017, pp. 1–11.

[8] L. Floridi and M. Chiriatti, "GPT-3: Its nature, scope, limits, and consequences," *Minds Mach.*, vol. 30, no. 4, pp. 681–694, Dec. 2020.

[9] D. Sobania, M. Briesch, and F. Rothlauf, "Choose your programming copilot: A comparison of the program synthesis performance of GitHub copilot and genetic programming," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2022, pp. 1019–1027.

[10] B. Adrian and U. S. R. Murty, *Graph Theory*. London, U.K.: Springer, 2008.

[11] R. Mina, C. Jabbour, and G. E. Sakr, "A review of machine learning techniques in analog integrated circuit design automation," *Electronics*, vol. 11, no. 3, p. 435, Jan. 2022.

[12] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. K. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar, "GNN-based hierarchical annotation for analog circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 9, pp. 2801–2814, Jan. 2023.

[13] K. Hakhamaneshi, M. Nassar, M. Phielipp, P. Abbeel, and V. Stojanovic, "Pretraining graph neural networks for few-shot analog circuit modeling and design," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 7, pp. 2163–2173, Oct. 2023.

[14] S. Yang, Z. Yang, D. Li, Y. Zhang, Z. Zhang, G. Song, and J. Hao, "Versatile multi-stage graph neural network for circuit representation," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 20313–20324.

[15] Y. Yamakaji, H. Shouno, and K. Fukushima, "Circuit2Graph: Circuits with graph neural networks," *IEEE Access*, vol. 12, pp. 51818–51827, 2024.

[16] D. Sánchez, L. Servadei, G. N. Kiprit, R. Wille, and W. Ecker, "A comprehensive survey on electronic design automation and graph neural networks: Theory and applications," *ACM Trans. Design Autom. Electron. Syst.*, vol. 28, no. 2, pp. 1–27, Mar. 2023.

[17] A. Hosny, S. Hashemi, M. Shalan, and S. Reda, "DRiLLS: Deep reinforcement learning for logic synthesis," in *Proc. 25th Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2020, pp. 581–586.

[18] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design (ICCAD)*, Nov. 2018, pp. 1–8.

[19] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, "ParaGraph: Layout parasitics and device parameter prediction using graph neural networks," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[20] X. Hong, T. Lin, Y. Shi, and B. H. Gwee, "GraphClusNet: A hierarchical graph neural network for recovered circuit netlist partitioning," *IEEE Trans. Artif. Intell.*, vol. 4, no. 5, pp. 1199–1213, Aug. 2023.

[21] D. S. Lopera, L. Servadei, G. N. Kiprit, S. Hazra, R. Wille, and W. Ecker, "A survey of graph neural networks for electronic design automation," in *Proc. ACM/IEEE 3rd Workshop Mach. Learn. CAD (MLCAD)*, Aug. 2021, pp. 1–6.

[22] W. Haaswijk, E. Collins, B. Seguin, M. Soeken, F. Kaplan, S. Süsstrunk, and G. D. Micheli, "Deep learning for logic optimization algorithms," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2018, pp. 1–4.

[23] H. M. Makrani, H. Sayadi, T. Mohsenin, S. Rafatirad, A. Sasan, and H. Homayoun, "XPPE: Cross-platform performance estimation of hardware accelerators using machine learning," in *Proc. 24th Asia South Pacific Design Autom. Conf.*, Jan. 2019, pp. 727–732.

[24] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nova, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean, "A graph placement methodology for fast chip design," *Nature*, vol. 594, no. 7862, pp. 207–212, Jun. 2021.

[25] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. 57th ACM/IEEE Design Autom. Conf. (DAC)*, Jul. 2020, pp. 1–6.

[26] A. B. Kahng, "New directions for learning-based IC design tools and methodologies," in *Proc. 23rd Asia South Pacific Design Autom. Conf. (ASP-DAC)*, Jan. 2018, pp. 405–410.

[27] W. Cao, M. Benosman, X. Zhang, and R. Ma, "Domain knowledge-infused deep learning for automated analog/radio-frequency circuit parameter optimization," in *Proc. 59th ACM/IEEE Design Autom. Conf.*, New York, NY, USA, Jul. 2022, pp. 1015–1020.

[28] Z. Wu and I. Savidis, "Transfer of performance models across analog circuit topologies with graph neural networks," in *Proc. ACM/IEEE 4th Workshop Mach. Learn. CAD (MLCAD)*, Sep. 2022, pp. 159–165.

[29] Y. Li and Y. W. Li, "Power converters topological transformation using dual and isomorphic principles," *IEEE Open J. Power Electron.*, vol. 1, pp. 74–87, 2020.

[30] R. Marquez and M. A. Contreras-Ordaz, "The three-terminal converter cell, graphs, and generation of DC-to-DC converter families," *IEEE Trans. Power Electron.*, vol. 35, no. 8, pp. 7725–7728, Aug. 2020.

[31] M. Liserre, V. Raveendran, and M. Andresen, "Graph-theory-based modeling and control for system-level optimization of smart transformers," *IEEE Trans. Ind. Electron.*, vol. 67, no. 10, pp. 8910–8920, Oct. 2020.

[32] Y.-S. Huang and J. R. Jiang, "Circuit learning: From decision trees to decision graphs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 42, no. 11, pp. 3985–3996, Mar. 2023.

[33] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. Int. Conf. Learn. Represent.*, 2016, pp. 1–14.

[34] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–12.

[35] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

[36] Z. Tong, Y. Liang, C. Sun, X. Li, D. Rosenblum, and A. Lim, "Digraph inception convolutional networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 17907–17918.

[37] X. Zhang, Y. He, N. Brugnone, M. Perlmutter, and M. Hirn, "MagNet: A neural network for directed graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 27003–27015.

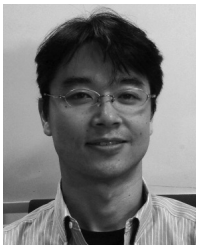
[38] E. Rossi, B. Charpentier, F. D. Giovanni, F. Frasca, S. Günnemann, and M. M. Bronstein, "Edge directionality improves learning on heterophilic graphs," in *Proc. 2nd Learn. Graphs Conf.*, Bertrand Charpentier, France, 2023, pp. 1–25.

[39] M. Engelhardt, "LTspice IV help file," Linear Technol. Corp., Milpitas, CA, USA, Tech. Rep., 2014. [Online]. Available: https://www.mdpi.com/2224-2708/5/4/15

[40] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch geometric," in *Proc. ICLR Workshop Represent. Learn. Graphs Manifolds*, 2019, pp. 1–9.

[41] Z. Zheng, X. Zhang, Y. Wang, S. He, C. Huang, L. Li, and D. Guo, "Classification of analog circuits based on graph convolution network," in *Proc. IEEE 16th Int. Conf. Anti-Counterfeiting, Secur., Identificat. (ASID)*, Dec. 2022, pp. 1–5.
[42] A. Said, M. Shabbir, B. Broll, W. Abbas, P. Völgyesi, and X. Koutsoukos, "Circuit design completion using graph neural networks," *Neural Comput. Appl.*, vol. 35, no. 16, pp. 12145–12157, Jun. 2023.
[43] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015. [Online]. Available: https://dare.uva.nl/search?identifier=a20791d3-1aff-464a-8544-268383c33a75
[44] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[45] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

YUSUKE YAMAKAJI received the B.Eng. and M.Eng. degrees in electrical and electronic engineering from the Institute of Science Tokyo (renamed from Tokyo Institute of Technology in October 2024), Tokyo, Japan, in 2009 and 2011, respectively. He is currently pursuing the Ph.D. degree in informatics and engineering with The University of Electro-Communications, Tokyo. His M.Eng. thesis was on the near field (multiple-pole expansion in the time domain) of antennas. In 2011, he joined Mitsubishi Electric Corporation, Kamakura, Japan. His research interests include AI, power integrity, and electromagnetic compatibility. As the primary inventor, he has received 17 patents in these fields.



HAYARU SHOUNO (Member, IEEE) received the Ph.D. degree in engineering from Osaka University, Osaka, in 1999. He is currently a Professor with the Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo, Japan. His research interests include computer vision, machine learning, and neural networks. He is an Elected Governor of the Asia Pacific Neural Network Society (APNNS). He is an Action Editor of *Neural Networks*.

KUNIHIKO FUKUSHIMA (Member, IEEE) received the B.Eng. degree in electronics and the Ph.D. degree in electrical engineering from Kyoto University, Kyoto, Japan, in 1958 and 1966, respectively. He was a Professor with Osaka University, Toyonaka, Japan, from 1989 to 1999; The University of Electro-Communications, Chofu, Japan, from 1999 to 2001; and Tokyo University of Technology, Hachioji, Japan, from 2001 to 2006. Furthermore, he was a Visiting Professor with Kansai University, Takatsuki, Japan, from 2006 to 2010. Prior to his professorship, he was a Senior Research Scientist with the NHK Science and Technology Research Laboratories, Setagaya, Japan. He is currently a Senior Research Scientist with the Fuzzy Logic Systems Institute (part-time), Iizuka, Japan, and works from home in Tokyo. He received the Achievement Award, the Distinguished Achievement and Contributions Award, the Excellent Paper Awards from IEICE, the Neural Networks Pioneer Award from IEEE, the APNNA Outstanding Achievement Award, the Excellent Paper Award, the Academic Award from JNNS, the INNS Helmholtz Award, the Pioneer Award from ELM2017, the Kenjiro Takayanagi Award 2019, and the Title of Distinguished Professor from the University of Electro-Communications. He was the Founding President of the Japanese Neural Network Society and a Founding Member of the Board of Governors of the International Neural Network Society. He is a former President of the Asia–Pacific Neural Network Assembly.