IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 19 February 2024, accepted 28 March 2024, date of publication 8 April 2024, date of current version 16 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3385862

RESEARCH ARTICLE

Circuit2Graph: Circuits With Graph Neural Networks

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 trial and error in both prototyping and simulation owing to the high degrees of freedom and mutual interference between the components. In this study, we propose a novel approach to address this challenge by introducing a transformation method that converts circuits into graph networks. This transformation is achieved with no loss of information, and we evaluate its accuracy through the application of graph classification by utilizing graph neural networks. We assume that the information degradation can result from self-loops, multi-edges, and heterogeneous graphs. To mitigate these issues, we propose a method that effectively reduces their impact. The results of this study demonstrate the effectiveness of our proposed method, as it achieves an accuracy of 97.89%. This represents a significant improvement of 5.2% when compared with the conventional method. Notably, our proposed method is applicable to general-purpose circuits. This makes it a valuable addition to the existing repertoire of circuit solution methods, alongside analytical and simulation approaches.

INDEX TERMS Circuit, graph neural network, ground node, homogeneous graph, one-hot embedding, star graph.

I. INTRODUCTION

Circuits form the foundation of electricity, playing a crucial role in its functioning. However, circuit enhancement requires extensive simulations and reliance on empirical rules. Therefore, empirical rules should be accelerated or an automation system should be established that surpasses them. To achieve this objective, the integration of circuits and neural networks is desirable. Neural networks are inspired by brain research [1]. By combining backpropagation [2] with large datasets, neural networks have outperformed humans in tasks involving datasets with coordinate systems, such as images and natural language processing [3]. For images, CNN-based [4], transformer-based [5], and MLP-based [6] methods outperform average human recognition and generation capabilities. In natural language processing, transformer-based [7] methods, exemplified by GPT-3 [8] and Copilot [9] have outperformed humans in tasks related to recognition and generation.

Circuits, however, lack spatial coordinates, such as images and natural language; therefore, they should be treated as graph networks that do not rely on a coordinate system [10], [11], [12]. Unlike continuous values in images and natural language, graph networks have discrete values that allow for a degree of freedom in the graph transformation method. Conventionally, circuit components and wiring are defined as nodes, with edges connecting these nodes [13], [14]. Because the roles of the circuit components and wiring differ, a constraint necessitates alternating appearances of component and wiring nodes. Unfortunately, training graph neural networks (GNNs) under these constraints poses challenges. The difficulty in satisfying these constraints results from the heterogeneous nature of the graph. Therefore, we propose a transformation to a homogeneous graph in

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

which the circuit components are nodes and the wiring components are edges.

To assess the effectiveness of the lossless transformation from circuit to graph (forward) and from graph to circuit (backward), it is crucial to demonstrate the electrical equivalence of these transformations. However, evaluating solely the forward transformation proves to be problematic owing to the inherent asymmetry between the forward and backward processes. To address this issue, we introduce a novel approach that utilizes GNN as shown in Fig.1. This method assumes that the preservation of information during the graph transformation results in reduced degradation and improved inference accuracy in GNN. The evaluation can focus only on the forward transformation provided this assumption is correct.

However, utilizing a homogeneous graph with circuit components as nodes and wiring as edges presents two significant challenges. The first problem involves multi-edges, particularly ground edges. In a graph network, ground edges lose their distinguishing feature as the reference potential because they carry the same information as other edges. In addition, ground edges generate numerous multi-edges, yet GNNs only focus on the presence or absence of edges, unable to distinguish between multi-edges and single edges. Therefore, we propose a solution wherein the ground is transformed into a node, serving as a circuit component. This method can significantly decrease the number of multi-edges while maintaining the homogeneity of the graph.

The second problem pertains to self-loops. When a circuit component with multiple terminals is transformed into a single node, the terminal numbers are lost, resulting in a self-loop representing a short circuit between the terminals of a single-circuit component. One potential transformation involves creating a complete graph, where each terminal becomes a node interconnected by edges [13]. However, complete graphs introduce two additional challenges. First, the computational complexity is increased. In a complete graph, the number of edges is proportional to the square of the number of terminals. This increase is acceptable for transistors with three or four terminals, as in previous studies, because the computational complexity of a GNN is proportional to the number of edges [15]. However, an exponential increase in computational complexity is unacceptable because a typical semiconductor has more than 10 terminals. Second, complete graphs exacerbate the issue of multi-edges, as short circuits between terminals are converted into multiple edges within the complete graph. However, this conversion is irreversible, and GNNs are unable to differentiate between single and multi-edges, resulting in information degradation. To address these issues, we propose the transformation of semiconductors into star graphs, that is, graphs in which the terminal nodes are connected through a single virtual node. This star graph representation allows a semiconductor with 1,000 terminals to be represented by 1,000 edges in a simple graph without self-loops or multi-edges, enabling a lossless transformation between the circuits and graphs.

When circuit constants are assigned to a graph, the graph structure and tabular data of the circuit constants for each node are managed separately [16]. However, this separation results in the inability to derive the circuit constants from the backward transformation, making it an irreversible process. Therefore, we propose the utilization of a one-hot embedding vector to assign circuit constants to node attributes. Although the method of assigning node attributes is arbitrary, they are one-hot vectors that hold orthogonality in the Adamar product [17]. In this one-hot embedding vector, a real number replaces the ‘‘1’’ in the one-hot vector. Because the dynamic range of the circuit constants exceeded 20 digits, we employed the logarithm of the circuit constant. The activation function in GNN responds to a range of ‘‘0 and 1’’ and ‘‘-1 and 1,’’ thereby normalizing the logarithmic values of the circuit constants (referred to as log-normalized values). This one-hot embedding vector transforms the circuit into a single graph and the circuit constants and log-normalized values into a linear mapping, facilitating a lossless transformation.

To validate our approach, we utilize a dataset of 3,308 sample circuits [18] provided by a circuit simulator [19] and evaluated them by utilizing graph classification. The main contributions of this study are as follows:

1) Homogeneous graph: It consists of circuit components and ground as nodes, with the wiring (excluding ground) serving as edges.
2) Star graph: Semiconductors are divided into terminal nodes, which are then connected through a virtual node.
3) One-hot embedding vector: The ‘‘1’’ element of the one-hot vector of node attributes is replaced by a real number that is proportional to a circuit constant.

A. PREVIOUS WORK

The social demand for automated circuit design has increased and GNNs have emerged as a promising solution for semiconductor applications [11], [12], [13], [14], [15], [16], [17], [18], [19]. The utilization of GNNs is driven by the increasing difficulty in adhering to stringent design rules, coupled with an increase in design cost owing to miniaturization and adoption of low-voltage transistors [11], [20]. For example, commercial electronic design automation tools provided by Synopsys Inc., Cadence Inc., Mentor Inc., and Zuken Inc. are employed for simulation, logic design, and analog design. However, these tools, which rely on design rules and simulations, require technical skill and incur simulation costs for parameter searching. Therefore, previous design data are required to mitigate these burdensome requirements and costs. Examples include the integration of logic synthesis [21], [22], [23], reinforcement learning for field-programmable gate array performance [21] and transistor placement [16]. Notably, these examples do not necessarily consider the circuit and graph reversibility

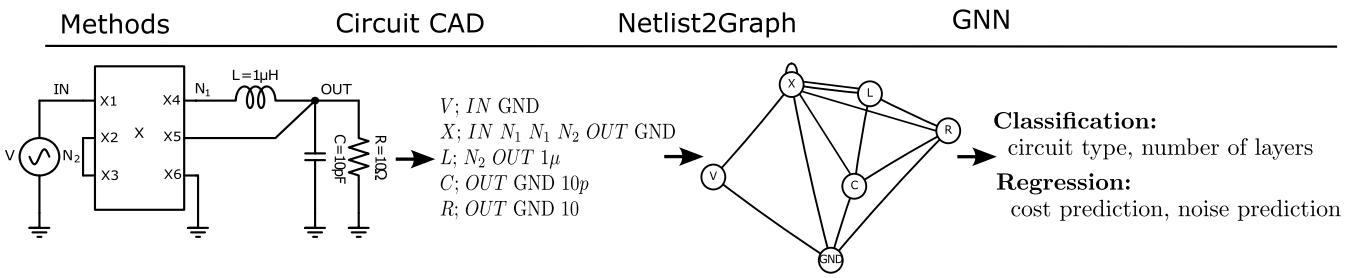


FIGURE 1. Netlist generation from the circuit, netlist transformation into a graph network, and evaluation of information degradation by graph classification accuracy.

because the output consists of images of the component layout [16] and numerical values for the simulation [13]. A method similar to the one-hot embedding vector concatenates the real values of the circuit constants into a one-hot vector [24], [25], [26]. However, these methods are not suitable for GNNs because they are not orthogonal, whereas the one-hot embedding vector is orthogonal between the circuit components.

In addition, previous studies have focused on the internal design of semiconductors, rendering them inapplicable to general-purpose electrical circuits, such as printed circuit boards and electric wiring in factories. In particular, many previous studies [24], [25] assumed that the internal structure and properties of semiconductors are known in the design phase. However, as regards their utilization, such as in the design of printed circuit boards, the internal structures are not publicly available and are treated as a black box. Similarly, when multiple devices are combined, such as in the electrical wiring of a factory, the characteristics of each device become complex and impossible to determine. Treating these devices and semiconductors as black boxes means considering them as nodes having only circuit components and no circuit constants. By approaching the circuit as a graph, we can manage circuit components with unknown characteristics, a capability that analytical solutions or simulations cannot achieve. For example, in printed circuit board design, component manufacturers publish block diagrams of semiconductors. However, the characteristics and circuit constants of each block in the block diagram are not published. Further, circuit constants of parasitic components, such as parasitic capacitance, residual inductance, mutual inductance, and residual resistance generated by the physical dimensions inside the component. Therefore, accurately determining these characteristics through actual measurements or simulations becomes challenging, necessitating the treatment of some or all semiconductors as black boxes. GNNs enable the treatment of components as black boxes that cannot be handled by circuit simulation.

II. GRAPH TRANSFORMATION

In logic circuits, transforming a circuit into a graph involves replacing each terminal of the transistors with a node and connecting these nodes with edges [27], [28]. This method connects the emitter, collector, and base of a transistor with three edges, resulting in a graph that is free of self-loops and multi-edges. However, when dealing with typical semiconductors with more than 10 terminals, the number of edges increases exponentially with the number of terminals. Another approach is to create a homogeneous graph where components serve as nodes and wirings act as edges. In this method, the ground is represented by edges since it functions as wiring. However, because most components operate with respect to the ground, the ground wiring generates multi-edges that are indistinguishable from other wiring types, resulting in a loss of connection between the circuit components and the ground. Furthermore, GNNs focus only on the presence or absence of edges between nodes and cannot consider the number of edges. This limitation makes multi-edges a degradation factor. Therefore, we propose a definition for the ground node, which allows any circuit to fit general GNNs, such as GCN [29] without modifying the underlying algorithms. In the star graph, each terminal node is connected through a virtual node, and a short circuit between the terminal nodes becomes an edge that directly connects them. In addition, because the computational complexity of GNN is proportional to the number of edges, the star graph enables GNN to operate efficiently, in terms of time and memory, even for large circuits with multiple semiconductors.

Moreover, to incorporate the circuit constants into the graph, we utilize a single embedding vector that can represent both the circuit component type and circuit constant simultaneously. This approach allows for seamless forward and backward transformations with no information degradation or loss of orthogonality. An example of circuit-to-GNN transformation is shown in Fig.2. The first step involves the addition of ground nodes and circuit constants, followed by the star graph transformation.

A. GROUND NODE

This section examines the ground node of the circuit shown in Fig.3. The netlist generated from the circuit is as follows:

V; IN GND
X; IN N_1 N_1 N_2 OUT GND
L; N_2 OUT 1μ
C; OUT GND $10p$
R; OUT GND 10

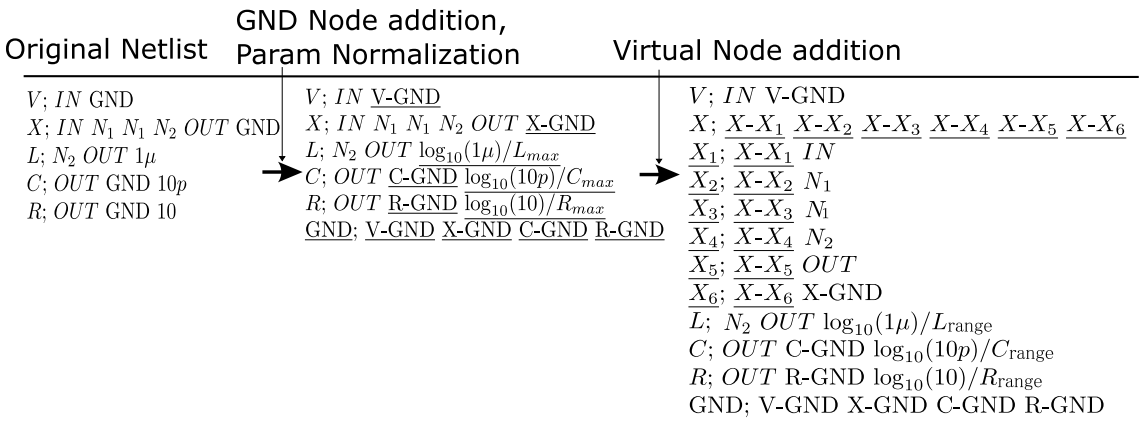


FIGURE 2. Netlist Update: The components are placed before the ";" and the wirings are placed after the ";." The first update involves making the ground a circuit component and log-normalizing the circuit constants. The second update converts semiconductors into star graphs.

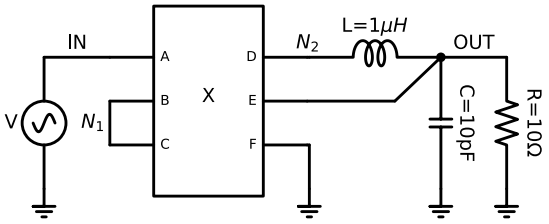


FIGURE 3. Circuit diagram: Semiconductor "X" has six terminals(X1-X6). X1 is connected to the power supply "V", X2 and X3 are short-circuited, X4 is connected to "L," X5 is connected to "R," "L" and "C" are feedback, and X6 is grounded.

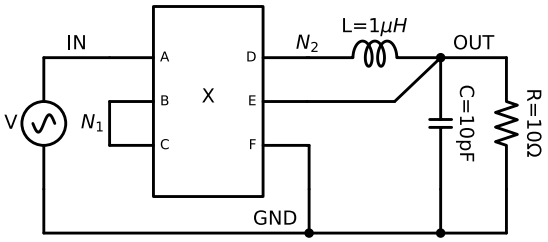


FIGURE 4. Circuit created from netlist without considering "GND" as the reference potential.

Netlists have over ten different formats, including Allegro, Telesis, and Scicards, which are mutually lossless transformations. Therefore, this study utilized the aforementioned netlist, which comprises only the components necessary for explanation. In this netlist, "V," "X," "L," "C," and "R" before ";" indicate the type of components, which are power supply, semiconductor, inductor, capacitor, and resistor, respectively. The elements after ";" represent wiring names and a circuit constant. "L," "C," and "R"' are passive elements and have two wiring names corresponding to both ends and a circuit constant in the third element. For example, "L" represents a 1μH coil connected to two wirings "N_2" and "OUT." "V" is the power supply, with wiring names for both ends. "X" has six wirings equal to the number of terminals and has a self-loop with wiring "N_1" connected between the second and third terminals. The wiring name "OUT", for example, indicates that the fifth terminal of "X" is connected to the second terminal of "L", the first terminal of "C" and the first terminal of "R," respectively. Also, between "X" and "L", there is a multi-edge connected by "N_2" and "OUT".

The circuit generated from the aforementioned netlist without the "GND" wiring as the reference potential is shown in Fig.4. The "GND" wiring is the same as the other wiring, resulting in edges between "V," "X," "C" and "R" connected by the "GND". Because the ground edges cannot be distinguished from the other edges, it becomes impossible to reconstruct a circuit from the graph.

However, the addition of a ground node is necessary for defining the new wiring between the circuit components connected to the ground node. For example, the wiring name between the nodes "V" and "GND" defines "V-GND"; the aforementioned netlist becomes:

V; IN V-GND
X; IN N_1 N_1 N_2 OUT X-GND
L; N_2 OUT $\log_{10}(1\mu)/L_{range}$
C; OUT C-GND $\log_{10}(10p)/C_{range}$
R; OUT R-GND $\log_{10}(10)/R_{range}$
GND; V-GND X-GND C-GND R-GND

The underlined text shows the modified points for clarity. Because the minimum and maximum ranges differ for each "R," "L," and "C," the ranges "R_{range}," "L_{range}" and "C_{range}" can also be used for each component. However, for clarity, we assumed that they have the same range as in this study. This netlist can reconstruct a circuit by assigning node attributes to the ground nodes in a one-hot embedding vector. The graph transformed from the updated netlist, comprising ten edges and a self-loop at "X" is shown in Fig.5. For comparison, a graph obtained from previous studies with the ground as an edge is shown in Fig.6. This graph has 14 edges, a self-loop at "X" and several multi-edges. Notably, GNN cannot consider the self-loop and multi-edges because it

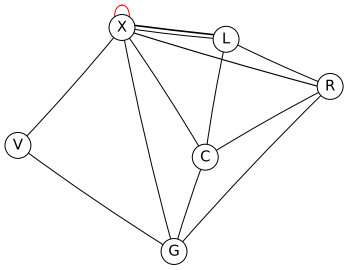


FIGURE 5. Homogeneous graph with the addition of the ground node. A graph with circuit components and ground as nodes in Fig.3.

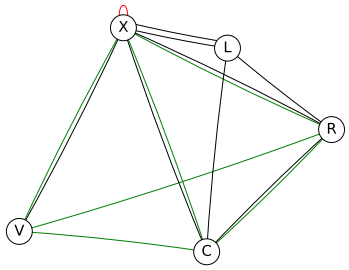


FIGURE 6. Homogeneous graph with the ground as edges. Green edges represent ground edges.

assigns self-loops to all nodes to pass their node attributes to the next hidden layer. Further, GNN cannot retain the terminal numbers necessary for the backward transformation.

B. STAR GRAPH OF SEMICONDUCTORS

The result of transforming a netlist into a graph based on [13] is shown in Fig.7. This method divides semiconductors into nodes for each terminal and creates a complete graph, eliminating self-loops. Consequently, the resulting graph does not have multi-edges, even for bus wiring between semiconductors. By replacing a semiconductor with a complete graph, short circuits between semiconductor terminals are converted into multi-edges. However, the bus wiring is represented without multi-edges. The advantages of dividing semiconductors outweigh the disadvantages because the number of short circuits decreases compared with bus wiring. Therefore, we utilize a portion [13] for comparison; a homogeneous graph with a ground node and a complete graph is shown in Fig.8. This transformation successfully addresses

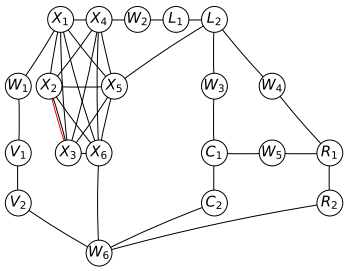


FIGURE 7. Heterogeneous graph based on [13]. A graph showing the terminals of all circuit components and wiring (W) as nodes.

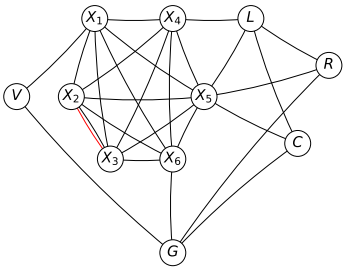


FIGURE 8. Homogeneous graph with a complete graph and a ground node, wiring as edges.

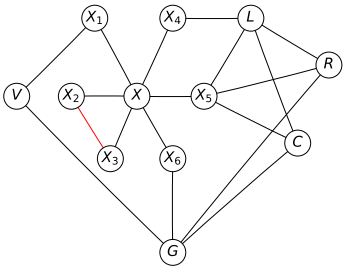


FIGURE 9. Homogeneous graphs with a star graph having a virtual node *X* to a semiconductor and terminal nodes connected through the virtual node.

the issue, except for the conversion of short circuits to multi-edges. To resolve this problem, we propose a semiconductor with a star graph and a virtual node at its center. This approach is shown in Fig.9, demonstrating the absence of multi-edges, even for self-loops caused by bus wiring or short circuits within a semiconductor. This graph is simple and highly reversible.

The following is a netlist of semiconductors divided into terminal nodes and connected by a newly defined virtual node: These changes are highlighted for clarity.

V; IN V-GND
X; $\underline{X\text{-}X_1}$ $\underline{X\text{-}X_2}$ $\underline{X\text{-}X_3}$ $\underline{X\text{-}X_4}$ $\underline{X\text{-}X_5}$ $\underline{X\text{-}X_6}$
$\underline{X_1}$; $\underline{X\text{-}X_1}$ IN
$\underline{X_2}$; $\underline{X\text{-}X_2}$ N_1
$\underline{X_3}$; $\underline{X\text{-}X_3}$ N_1
$\underline{X_4}$; $\underline{X\text{-}X_4}$ N_2
$\underline{X_5}$; $\underline{X\text{-}X_5}$ OUT
$\underline{X_6}$; $\underline{X\text{-}X_6}$ X-GND
L; N_2 OUT $\log_{10}(1\mu)/L_{range}$
C; OUT C-GND $\log_{10}(10p)/C_{range}$
R; OUT R-GND $\log_{10}(10)/R_{range}$
GND; V-GND X-GND C-GND R-GND

C. ONE-HOT EMBEDDING VECTOR

In the aforementioned netlist, the values of the inductor L, capacitor C, and resistor R are 1μH, $10p$F, and 10Ω, respectively. To ensure that the GNN recognizes the circuit constants, [16] proposed a method that combines the features of the GNN with those fully connected to the tabular-formed circuit constants. However, the circuit topology and circuit constants are maintained separately; therefore, this relationship cannot be maintained. Establishing a close

relationship between the circuit constants in the graph is crucial. The previous studies [25], [26] could not relate the component type and circuit constant simultaneously. To address these limitations, we proposed a one-hot embedding vector for the node attributes. The one-hot vectors shown in Fig.9 comprise six elements, "X," "V," "L," "C," "R" and "GND." Each node is represented as a matrix with one row and six columns. The one-hot embedding vectors for the "L," "C" and "R" with their circuit constants are listed in Table 1.

TABLE 1. One-hot embedding vector: The "1" element of a one-hot vector is replaced with a circuit constant.

X	1	0	0	0	0	0
X:1	1	0	0	0	0	0
X:2	1	0	0	0	0	0
X:3	1	0	0	0	0	0
X:4	1	0	0	0	0	0
X:5	1	0	0	0	0	0
X:6	1	0	0	0	0	0
V	0	1	0	0	0	0
L	0	0	$\mathbf{1}\mu$	0	0	0
C	0	0	0	$\mathbf{10}p$	0	0
R	0	0	0	0	**10**	0
GND	0	0	0	0	0	1

The range of the circuit constants R, L, and C is approximately 10^{22}. In particular, components close to the minimum value are zero, resulting in a one-hot embedding matrix with all zero elements. This results in a loss of information regarding the component type and circuit constants simultaneously. Therefore, as listed in Table 1, the real values of the circuit constants with logarithms were applied. Although the base of the logarithm was arbitrary, it was set to 10. Because the logarithm of the circuit constant of 1 becomes zero, the logarithm of the circuit constant of 1 is assigned a small value of 10^{-3}. Consequently, all circuit constants are linearly mapped to real numbers ranging from -13 to $+9$.

Because the activation function of GNN responds to a real number that ranges from 0 to 1 or -1 to 1, a linear mapping of -13 to $+9$ in this range should improve the inference accuracy of GNN. Therefore, the circuit constant $\mathbf{x}$ was log-normalized using eq.(1), between 0.07 and 0.98.

TABLE 2. One-hot embedding vector: the "1" element of the one-hot vector is replaced with the logarithm of the circuit constant.

X	1	0	0	0	0	0
X:1	1	0	0	0	0	0
X:2	1	0	0	0	0	0
X:3	1	0	0	0	0	0
X:4	1	0	0	0	0	0
X:5	1	0	0	0	0	0
X:6	1	0	0	0	0	0
V	0	1	0	0	0	0
L	0	0	**-6**	0	0	0
C	0	0	0	**-11**	0	0
R	0	0	0	0	**1**	0
GND	0	0	0	0	0	1

TABLE 3. One-hot embedding vector: node attributes with log-normalization eq.(1) applied to circuit constants.

X	1	0	0	0	0	0
X:1	1	0	0	0	0	0
X:2	1	0	0	0	0	0
X:3	1	0	0	0	0	0
X:4	1	0	0	0	0	0
X:5	1	0	0	0	0	0
X:6	1	0	0	0	0	0
V	0	1	0	0	0	0
L	0	0	**0.35**	0	0	0
C	0	0	0	**0.13**	0	0
R	0	0	0	0	**0.65**	0
GND	0	0	0	0	0	1

In the equation, the $\mathbb{A}$ means a specific set that takes a circuit constant when the type of circuit component and its position in the one-hot vector are coincident. The results obtained by updating Table 2 are listed in Table 3.

$$\mathbf{1}_{\mathbb{A}}(\mathbf{x}) = \begin{cases} \dfrac{\log_{10}(\mathbf{x})}{\max(\log_{10}(\mathbf{x})) - \min(\log_{10}(\mathbf{x})) + 1} + \\ \qquad \dfrac{1}{|\min(\log_{10}(\mathbf{x}))| + 1} & \text{if } \mathbf{x} \in \mathbb{A} \\ 0 & \text{if } \mathbf{x} \notin \mathbb{A} \end{cases} \tag{1}$$

Because activation functions, such as the Tanh function, respond to values between -1 and 1, the circuit constant is updated by utilizing eq.(2), assuming that it is combined with a Tanh function. Compared with eq.(1), the range is doubled and the inference accuracy is expected to improve owing to the improved rounding errors. The results obtained by updating Table 2 are shown in Table 4.

$$\mathbf{1}_{\mathbb{A}}(\mathbf{x}) = \begin{cases} \dfrac{\log_{10}(\mathbf{x})}{\max(\log_{10}(\mathbf{x})) - \min(\log_{10}(\mathbf{x}))} & \text{if } \mathbf{x} \in \mathbb{A} \\ 0 & \text{if } \mathbf{x} \notin \mathbb{A} \end{cases} \tag{2}$$

TABLE 4. One-hot embedding vector: node attributes with log-normalization eq.(2) applied to circuit constants.

X	1	0	0	0	0	0
X:1	1	0	0	0	0	0
X:2	1	0	0	0	0	0
X:3	1	0	0	0	0	0
X:4	1	0	0	0	0	0
X:5	1	0	0	0	0	0
X:6	1	0	0	0	0	0
V	0	1	0	0	0	0
L	0	0	**-0.46**	0	0	0
C	0	0	0	**-0.85**	0	0
R	0	0	0	0	**0.08**	0
GND	0	0	0	0	0	1

III. EXPERIMENTS

A. DATASET

The netlist generated from the circuit contains important design data and is generally not available to the public.

Previous studies have been conducted in collaboration with semiconductor manufacturers. The only publicly available datasets comprise only 60 circuits [30], which are insufficient for GNNs or particular circuits, such as ladder circuits and operational amplifiers(Op-amp) [13]. Moreover, these datasets are publicly available in graph network format; therefore, the transformation from a circuit to a graph network, which is the purpose of this study, cannot be evaluated. Therefore, we utilized circuits provided by LTspice [19], a free commercially available circuit simulator from Analog Devices Inc. comprising 3,308 distinct semiconductors with different model numbers. Datasets were classified into seven types based on the type of semiconductor utilized in the circuits. Out of the 3,308 circuits, 2,315 (70% of the total) were designated as training data, whereas the remaining 993 were utilized for testing. In general, the position of "1" in the one-hot vector can vary depending on the type of semiconductor, such as power supplies, op-amps, and switches. However, in this dataset, the types of semiconductors and correct answer labels were identical. Hence, the node attribute of a semiconductor corresponds directly to the semiconductor itself, as shown in Table 3 and Table 4. Notably, LTspice offered 14 types of circuits for correct answer labels; however, seven were excluded because the number of sample circuits was approximately 10, which was not suitable for training and inference.

The number of circuits representing the seven types of circuits in the dataset and the number of nodes for the following five different transformations to the graph network are shown in Table 5. Compared with 1) previous study [13], 2) and 3) have approximately 1/10 of the number of nodes, whereas 4) and 5) have approximately 1/4 of the number of nodes.

TABLE 5. Average number of nodes in each transformation of circuits.

Type (Number of Circuits)	ADC (36)	Comp. (41)	Filter Products (25)	Opamps (681)	Power Products (2340)	Ref. (66)	Switches (119)	Total (3308)
1) Nodes w/ Wire node [13]	151.28	48.29	150.80	46.58	145.93	32.57	54.09	118.25
2) Nodes w/o GND node	11.75	7.60	12.60	8.98	13.00	5.17	7.88	11.75
3) Nodes w/ GND node	12.75	8.60	13.60	9.98	14.00	6.17	8.88	12.75
4) Nodes w/ GND node +Complete Graph	29.56	15.56	27.32	16.78	27.37	11.63	15.75	24.29
5) Nodes w/ GND node +Star Graph	32.00	16.76	29.80	18.17	29.14	13.40	16.77	25.93

The numbers of edges under the same conditions as those listed in Table 5 are listed in Table 6. Compared with 1) previous study [13], the number of edges in 2) is less than half, 3) is approximately 1/4, 4) is approximately the same, and 5) is less than half. As the computational cost of a GNN is linearly or exponentially proportional to the number of edges, the ground node in 3) and star graphs in 5) are superior.

B. GNN PARAMETERS

Because of the statistical nature of GNNs, variations in the results should be considered. The inference accuracy of GNN depends on the distribution of the training and test data, GNN model, and hyperparameters of the GNN. Notably, factors other than the number of node attributes associated with the transformation remain fixed. In addition, we generated 10 datasets with different combinations of training and test data, and the inference accuracy was assessed by averaging their test accuracy.

TABLE 6. Average number of edges in each circuit transformation.

Type (Number of Circuits)	ADC (36)	Comp. (41)	Filter Products (25)	Opamps (681)	Power Products (2340)	Ref. (66)	Switches (119)	Total (3308)
1) Edges w/ Wire node [13]	145.64	47.49	147.08	46.04	145.64	31.71	53.88	117.82
2) Edges w/o GND node	73.17	21.61	36.80	25.35	60.48	14.30	40.78	50.93
3) Edges w/ GND node	23.75	19.56	32.92	21.22	37.78	11.45	15.65	32.63
4) Edges w/ GND node +Complete Graph	138.72	47.02	134.32	44.04	152.00	30.89	48.50	121.45
5) Edges w/ GND node +Star Graph	51.97	29.20	51.32	30.47	61.44	22.14	24.19	52.17

The inference accuracy was validated by utilizing five hyperparameters of the GNN. The maximum value of each channel in the hidden layer was passed through the two fully connected layers and the softmax function at the output. Each hidden layer of the GNN has activation and normalization functions, namely GraphNorm [31]. The mini-batch comprised 2,315 training data with a data size of less than 2 MB. The maximum GPU memory usage when training without mini-batches was 12 GB. Therefore, mini-batches were deemed unnecessary.

1) GNN algorithm: GraphSage [32] is best compared with GCN [29], GAT [33].
2) Number of hidden layers: From 3 to 5 are best; hence, we select three layers.
3) Activation function: Tanh is best compared with ReLU and LeakyReLU [34].
4) Hiden Layer Normalization: GraphNorm [31] is best compared with BatchNorm [35] and LayerNorm [36].

C. SUMMARY OF RESULTS

The experimental results are summarized in Table 7. The hyperparameters of the results are as follows:

1) Semiconductor: single, complete graph, star graph
2) Presence/Absence of ground node
3) Circuit constants: w/o, w/ (direct, log, "0 to 1," and "-1 to 1")
4) Model: GraphSage [32], GAT [33], GCN [29]
5) Activation Functions: ReLU, Tanh, and LeakyReLU [34]

The results obtained by applying previous studies to the dataset for comparison are listed in Table 7. However, [13] did not provide guidance on assigning circuit constants. Thus, we combined the features obtained by a fully connected layer with the circuit constants, as expressed in a tabular form [16]. The inference accuracy was 93.76% for the heterogeneous graphs [13], [16] in scenario (1). The homogeneous graph in scenario (2) improves the accuracy by over 3% compared with scenario (1). The accuracy further increased by 0.4% compared with scenario (2) by adding a ground node. Based on homogeneous graphs with a ground node, the highest inference accuracy was 97.89% in scenario (7), corresponding to an average accuracy improvement of 0.2% compared with the second-highest accuracy in scenario (5). Thus, star graphs without multi-edges and self-loops demonstrate

TABLE 7. 1. previous study [13], [16], 2. semiconductor with a single node (Single), as well as complete graph and star graph, 3. best GNN inference accuracy with and without a ground node.

	w/ Param	w/o Param
(1) Complete w/ Wire node [13] [16]	13th. 93.76±1.02 0 to 1(Tanh)	14th. 92.69±1.03 (Tanh)
(2) Single w/o GND node	10th. 97.01±0.61 0 to 1(Tanh)	11th. 97.00±0.54 (Tanh)
(3) Single w/ GND node	6th. 97.40±0.69 -1 to 1(Tanh)	9th. 97.25±0.56 (Tanh)
(4) Complete w/o GND node	7th. 97.32±0.52 -1 to 1(Tanh)	12th. 96.77±1.13 (ReLU)
(5) Complete w/ GND node	2nd. 97.68±1.75 -1 to 1(ReLU)	4th. 97.48±0.55 (ReLU)
(6) Star w/o GND node	5th. 97.48±0.46 log10(Tanh)	8th. 97.31±0.63 (LeakyReLU)
(7) Star w/ GND node	1st. **97.89±0.87** 0 to 1(Tanh)	3rd. 97.51±0.57 (ReLU)

superior performance in terms of lossless transformation and GNN inference accuracy. The Training was performed on a computer with a CPU: Intel Xeon Gold 5115 and a GPU: NVIDIA QuadroRTX8000. The training time for scenario (2) amounted to 350 s, whereas scenario (3) required 300 s, resulting in a notable 20% reduction in computation time. In comparison, the training time for complete graphs was approximately twice as long as that of star graphs at 950 s and 400 s, respectively. These results were proportional to the number of edges, as listed in Table 6.

D. ONE-HOT EMBEDDING VECTOR

To evaluate the differences between various GNN models, we present the outcomes for GraphSage [32], GAT [33], and GCN [29] with ground node and one-hot embedding vectors when the semiconductor is transformed into a star graph. As shown in Table 8, the ground node and one-hot embedding vector both contributed to the inference accuracy, regardless of the GNN model.

TABLE 8. Differences in inference accuracy of GNNs owing to different methods of assigning circuit constants to node attributes.

	One-Hot Value	w/ GND node	w/o GND node
GraphSage	1:w/o Param	97.51±2.49(ReLU)	97.31±0.63(ReLU)
	Anti-log	73.37±2.35(ReLU)	71.25±3.44(ReLU)
	log10	97.51±0.64(Tanh)	97.48±0.46(Tanh)
	0 to 1	**97.89±0.87(Tanh)**	97.16±0.58(Tanh)
	-1 to 1	97.65±1.05(Tanh)	97.11±0.92(LeakyReLU)
GAT	1:w/o Param	96.70±1.42(Tanh)	95.62±0.37(ReLU)
	Anti-log	73.22±6.31(Tanh)	81.13±12.85(Tanh)
	log10	96.70±0.42(LeakyReLU)	96.26±0.68(LeakyReLU)
	0 to 1	97.05±0.58(ReLU)	96.20±0.52(Tanh)
	-1 to 1	96.98±0.88(Tanh)	96.31±1.15(Tanh)
GCN	1:w/o Param	96.72±0.73(LeakyReLU)	95.89±0.73(ReLU)
	Anti-log	78.02±3.32(LeakyReLU)	76.80±2.01(LeakyReLU)
	log10	96.87±0.47(Tanh)	96.22±0.49(ReLU)
	0 to 1	96.61±0.72(LeakyReLU)	96.22±1.47(Tanh)
	-1 to 1	96.89±1.01(ReLU)	96.06±0.65(Tanh)

E. DISCUSSION

Assuming the inference accuracy would improve with minimal degradation of information during the transformation into graphs, we presented the circuit and graph transformation methods in terms of reversibility. We hypothesized that the inference accuracy of GNNs would increase in the order of circuit and graph reversibility. As shown in Fig.7, the hypothesis is valid because it holds true regardless of the GNN model. Furthermore, GraphSage outperforms GCN and GAT by over 0.5%, as shown in Fig.7. This is because GraphSage utilizes only randomly sampled nodes. In contrast, GCN and GAT utilize all the nodes in each hidden layer and consider factors, such as power supply and load resistance, which are not necessary for our graph classification.

Although this study focused on graph classification, our graphs can handle general-purpose circuits, making them potentially applicable to the following six areas, provided the appropriate GNN model and dataset are employed. The characteristics of the circuit graph that utilizes GNNs are shown in Fig.9.

1) Graph classification: classification of normal and abnormal circuits, predicting the number of layers of boards and semiconductors required for assembly.
2) Graph regression: Prediction of total equipment cost and board area required for assembly.
3) Link Prediction: Prediction of the presence or absence of wiring between components.
4) Node Prediction: Component selection and circuit constant optimization.
5) Graph Autoencoder (unsupervised training): Detection of similar circuits based on circuit features.
6) Graph Decoder (unsupervised learning): Circuit generation with desired characteristics.

Notably, our two problems could not be resolved using our method: directional components, such as diodes, and magnetic coupling, which has no wiring and is represented by spatial coupling. The dataset in this study contained 1,791 diodes in 893 circuits; however, they were treated as non-directional components in GCN, GAT and GraphSage. Further, the dataset contained 131 single or three-phase magnetic circuits. However, magnetic coupling cannot be represented by wiring or components; therefore, it is lost during our graph transformation. As 2.11% of the errors within the 97.89% inference accuracy included circuits with diodes and magnetic couplings, solving these problems could improve the inference accuracy. These issues should be addressed in future studies. However, GNN is based on statistical processing, and even with future improvements, it will not be able to solve 100% of all circuits. Therefore, depending on the problem to be solved, GNN should be used in combination with circuit simulation.

TABLE 9. Comparison of the characteristics of the conventional analytical method, simulation, and our circuit solution using GNN.

	Strictness	Calculation Cost	Target	Number of Components
Analytical	Very Good	Bad	Impedance	<10
Simulation	Good	Good	Voltage/Current	<1,000
GNN	Bad	Very Good	Multiple	<100,000,000

IV. CONCLUSION

This study proposed a method to mitigate information degradation during the reversible transformation from a circuit into a graph network. We demonstrated that heterogeneous graphs resulted in degradation, whereas homogeneous graphs could mitigate information degradation. Furthermore, we identified self-loops and multi-edges as factors that contributed to degradation in homogeneous graphs. To mitigate this effect, we proposed the incorporation of the reference potential (ground) as a node and semiconductors as star graphs. Furthermore, we introduced a one-hot embedding vector that replaced the "0 to 1" of the one-hot vector of node attributes with a real number proportional to the circuit constant. To validate the effectiveness of our proposed methods, we conducted a comprehensive graph classification study involving 3,308 circuits categorized into seven types. Our proposed method achieved an accuracy of 97.89%, surpassing that of previous studies by over 5%.

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] Y. Lecun, "A theoretical framework for back-propagation," in *Proc. Connectionist Models Summer School*, 1988, pp. 21–28.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, Sep. 2020.

[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," 2020, *arXiv:2010.11929*.

[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 *Advances in Neural Information Processing Systems*, 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 *Advances in Neural Information Processing Systems*, vol. 30. Red Hook, NY, USA: Curran Associates, 2017.

[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] J. A. Bondy and U. S. R. Murty, *Graph Theory*. London, U.K.: Springer, 1976.

[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, Sep. 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, May 2022.

[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] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[16] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, 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.

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

[18] 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.

[19] M. Engelhardt, "LTSPICE IV help file," Linear Technol. Corp., Milpitas, CA, USA, 2014.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[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. K. Khamis and M. Agamy, "Comprehensive mapping of continuous/switching circuits in CCM and DCM to machine learning domain using homogeneous graph neural networks," *IEEE Open J. Circuits Syst.*, vol. 4, pp. 50–69, 2023.

[27] 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.

[28] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High performance graph Convolutional networks with applications in testability analysis," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*.

[30] 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.

[31] T. Cai, S. Luo, K. Xu, Di He, T.-Y. Liu, and L. Wang, "GraphNorm: A principled approach to accelerating graph neural network training," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 1204–1215.

[32] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, vol. 30. Red Hook, NY, USA: Curran Associates, 2017.

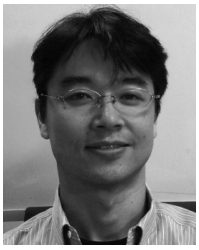
[33] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," 2017, *arXiv:1710.10903*.

[34] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," 2015, *arXiv:1505.00853*.

[35] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[36] J. L. Ba, J. Ryan Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.

YUSUKE YAMAKAJI received the B.E. and M.S. degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2011. In 2011, he joined Mitsubishi Electric Corporation, Kamakura, Japan. His research interests include AI, power integrity, and electromagnetic compatibility. He holds more than 15 patents. He is currently pursuing the Ph.D. degree with the University of Electro-Communications, Tokyo, Japan, while working at Mitsubishi Electric Corporation.



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. His research interests include the field of computer vision and machine learning, including neural networks. He is an Action Editor of *Neural Networks* and an Elected Governor of the Asia–Pacific Neural Network Society (APNNS).

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. 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. 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 Japanese Neural Network Society (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 JNNS and was a Founding Member on the Board of Governors of the International Neural Network Society. He is a former President of the Asia–Pacific Neural Network Assembly.