IEEE Open Journal of
Power Electronics

Received 24 May 2024; accepted 10 June 2024. Date of publication 18 June 2024;
date of current version 3 July 2024. The review of this article was arranged by Associate Editor T. B. Lazzarin.

Digital Object Identifier 10.1109/OJPEL.2024.3416195

Circuit Dynamics Prediction via Graph Neural Network & Graph Framework Integration: Three Phase Inverter Case Study

AHMED K. KHAMIS [1,2] AND MOHAMMED AGAMY [1] (Senior Member, IEEE)

[1]ECE Department, University at Albany, SUNY Albany, NY 12222 USA
[2]EE Department, AASTMT, Alexandria 21937, Egypt

CORRESPONDING AUTHOR: AHMED K. KHAMIS (e-mail: ahmedkhamis47@gmail.com)

ABSTRACT This article proposes an integration between a graph framework for circuit representation and a Graph neural network (GNN) model suitable for different machine learning (ML) applications. Furthermore, the paper highlights design steps for tailoring and using the GNN-based ML model for converter performance predictions based on converter circuit level and internal parameter variations. Regardless of the number of components or connections present in a converter circuit, the proposed model can be readily scaled to incorporate different converter circuit topologies and may be used to analyze such circuits regardless of the number of components used or control parameters varied. To enable the use of ML methods and applications, all physical and switching circuit properties including operating mode, components and circuit behavior must be accurately mapped to graph representation. The model scalability to other circuit types and different connections and circuits elements is also tested, while being studied in the most common DC-AC inverter in grid connected systems including filter and filterless configurations. The filtered and filterless DC-AC inverter circuits are used to evaluate the model, scoring R^2 greater than 99% in most cases and a mean square error (MSE) tending to zero.

INDEX TERMS Electric circuit, bond graph, graph neural networks (GNN), machine learning (ML).

I. INTRODUCTION

Artificial intelligence has been incorporated in the form of deep learning (DL) models in a wide range of disciplines. In particular, the use of recurrent neural networks (RNN) for sequential processing and convolutional neural networks (CNN) in electrical and renewable energy applications has been gaining momentum [1], [2]. Recently, graph neural networks (GNN), which model patterns in graph-structured data, have seen a surge in popularity; these networks are particularly advantageous for representing electrical circuit structure, as graphs are a natural data format for expressing such information.

In [3], GNNs were proposed as suitable alternatives to shallow methods or mathematical optimization techniques for circuit optimization/classification needs and multiple applications (e.g., transistor sizing, capacitor value optimization), with the general outline in Fig. 1. [4], [5] used a reinforcement learning (RL) agent to select optimal parameters via rewarding based on figure of merit (FOM) when circuits were represented as graphs (nodes/edges refer to components/wires, each transistor embedded with a vector). Ref. [6] used differential neural network (DNN) for mapping a circuit to its corresponding transfer function, but applicable only for a specific topology. Ref. [7] combined feature maps of nodes via GNN to simulate a distributed circuit's electromagnetic properties. Ref. [8] used DeepGEN for predicting ladder and two-stage operational amplifier circuits with up to 10 branches, but lacked description of connection type and other elements, e.g., frequency, phase shift. Ref. [9] used GNN to identify symmetry constraints in analog circuits and proposed extending it to other constraints. Refs. [10] represented elements as heterogeneous multi-graphs and set four types of edges. Refs. [3], and [4], [5] used graph models to represent circuits, and [6], [7], [8], [9], [10] leveraged GNNs for circuit optimization/classification needs, RL-based selection of optimal parameters, transfer function mapping,

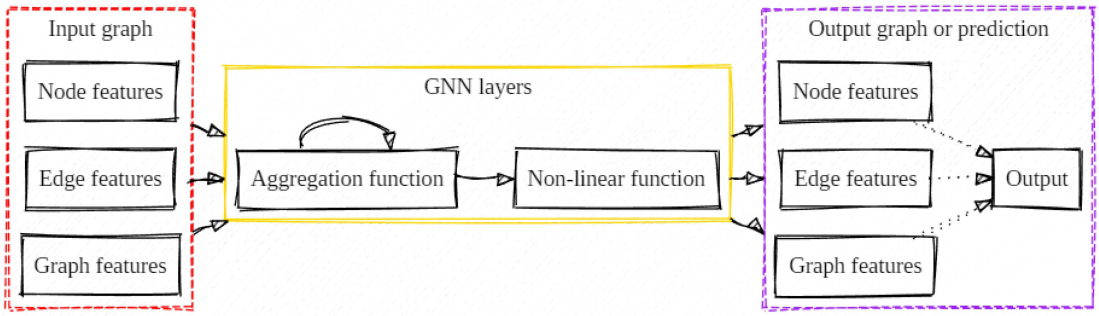

FIGURE 1. GNN general architecture and operation principle.

electromagnetic simulation and symmetry constraints identification, respectively. GNNs are not only capable of quickly training on graphs, but also generalizing to large datasets, and learning order permutation invariant representations from the graph modeling approaches, but they have also been applied to circuit design [7], [11], [12], though structure-based predictions in switching converter circuits have yet to be addressed. In [13], a comparative review of different research attempts in mapping circuits to ML domain including circuit representation techniques was presented. Additionally, the three possible circuit representation techniques listing their advantages and disadvantages were highlighted, while proposing a graph framework for representing electric circuit as graphs with unified node and edge features assignment that is generalizable to continuous and switching circuits alike. Moreover, a dataset generation algorithm capturing circuit performance in a standardized data format was introduced, allowing for the the training of any ML model using circuit graph data. Furthermore, a classifier problem applied to multiple converter topologies including resonant and DC-DC converters in CCM or DCM was presented. In this paper the same GNN based framework is leveraged and used for regression ML task, including performance evaluation of the model behavior when subjects to severe input variations like topology and parameter variations. Moreover, a study of the ability of the proposed graph framework to accommodate such severe variations and their effect on scalability and usability as well as on time and space complexity of the GNN based model is introduced. From applications point of view, the study offers and explain how to utilize the graph framework for hot and interesting topics like AI circuit generation and real time parameter estimation. Finally, the study shows some limitations that affects the performance, with some open research tracks that can be addressed in the future.

II. SIMULATORS & ANALYTICAL MODELS VS ML MODELS

ML models can be extremely useful to predict the solution of a converter behavior when simulators or analytical models are not sufficient/beneficial. Some of the reasons why simulators or analytical models may have limitations are:

1) Simulators or analytical models are too complex or computationally expensive to run, especially for large-scale or high-dimensional systems (high circuit order, number of components... etc). ML models can act as emulators or surrogate models that can approximate the behavior of the system with much less computational cost and time.
2) Simulators or analytical models may rely on simplifying assumptions by designer that may not hold in reality. Additionally, they may not account for all the nonlinear, dynamic, stochastic, and complex phenomena that occur in real-world systems, or require a lot of computational resources, data, and parameters that may not be easily obtained or varied. However, ML models can learn from data and incorporate prior knowledge or physical laws to improve the accuracy and robustness of the predictions.

When utilizing ML models and integrating it with the proposed framework for circuit prediction, the following benefits are gained:

- *Speed and scalability:* Proposed ML model combined with proposed graph framework can provide fast and accurate instant predictions for a wide range of circuit topologies and parameters, without requiring complex mathematical derivations or simulations [14], [15]. This point will be discussed in details in Section VI
- *Flexibility and adaptability:* ML models can handle different control schemes, component variations, and operating conditions without changing the circuit structure or adding components to change the control scheme. This is not the case in analytical models, where the analytical model has to be altered on controller components are to be added to simulate a change in control scheme [15].
- *Generalization and transferability:* ML models can learn from a limited number of training examples and generalize to unseen data, and also transfer the learned knowledge to other related tasks or domains [16], [17].
- *Integration of multi-domain physics:* The proposed graph framework is based on bond graph representation, allowing for integration of different physical domains like mechanical, chemical, and electrical domains and properties when representing electric circuits [18], [19], [20], [21].

The following contributions are proposed and addressed in this paper as:

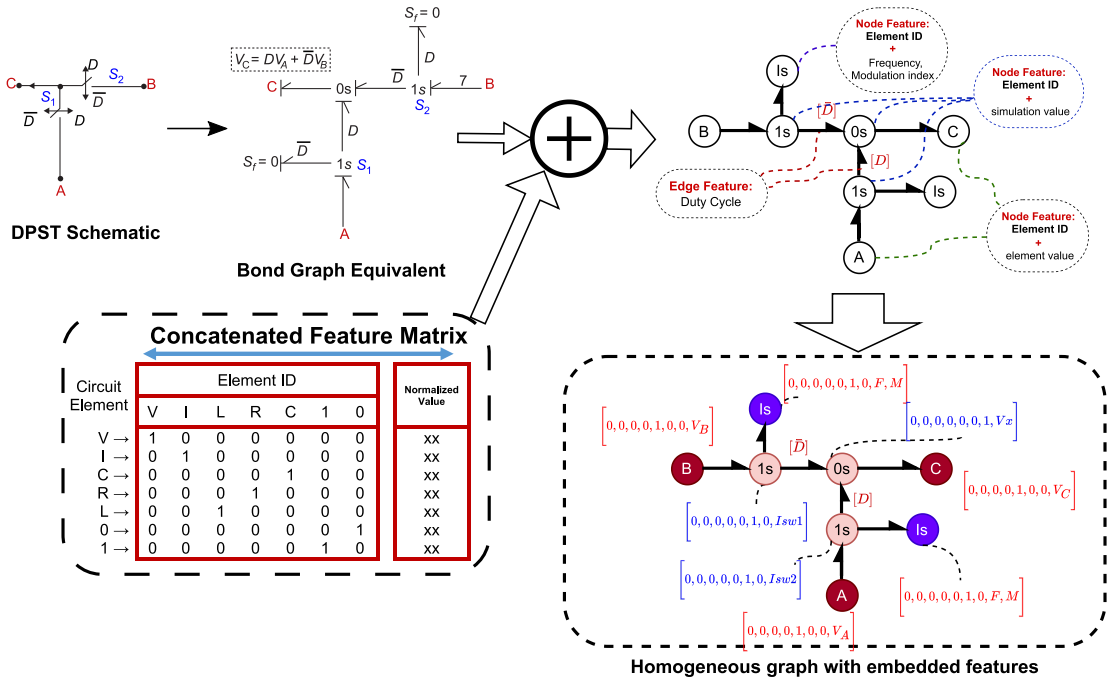

FIGURE 2. Switching cell and its bond graph representation.

- Providing a systematic and a mathematical formulation for building *any* GNN model (regardless or the model layers, number of neurons or type of GNN used) intended for circuit dynamics prediction tasks or any other ML tasks.
- An investigation of the generalization and applicability of the graph framework and model performance and its adaptability to other circuit variations like:
  - Multi-variable dynamics prediction tasks.
  - Variation in circuit structure.
  - Circuit components, input source, output load variations.
  - Controller modulation and control scheme variations.
- A mathematical breakdown of the time and space complexities in terms of number of nodes and features, allowing for the calculation of the running time and memory needed and the predetermination of required hardware for running the GNN model.
- A statistical analysis of the model behavior including error, fitting analysis and the model generalization capability.
- An experimental validation of the proposed framework accuracy and usability implemented on a physical three-phase DC-AC converter circuit.

III. CIRCUIT REPRESENTATION IN ML DOMAIN

This work proposes converter dynamics predictions based on the physical connection and operating circumstances of a converter, based on circuits to machine learning (ML) domain mapping approach published in [13], [22], [23]. Bond graph modeling with switching circuit representation is used to transform circuits to graphs, from which a graph dataset is created.

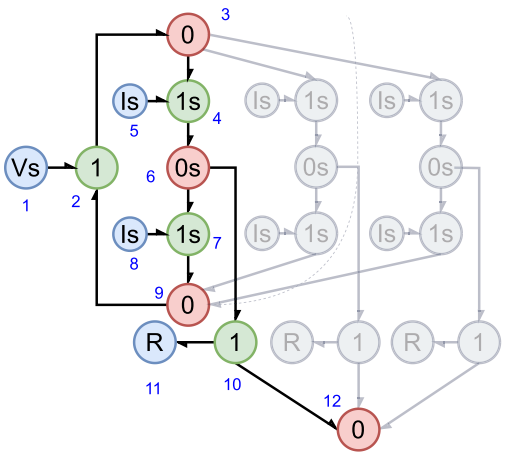

FIGURE 3. One-leg of the three-phase inverter bond graph in Fig. 5(b).

A. BOND GRAPH FOR GRAPHICAL CIRCUIT REPRESENTATION

Electric circuits are to be modelled using bond graph circuit representation, in which junctions represent connections that connect to components and energy is exchanged between system components and junctions. One-junctions represent series connections, where the flow variable (e.g. current in an electrical circuit) is the same for all connected components while the effort variable (e.g. current in an electrical circuit) is conserved. On the other hand, zero-junctions represent parallel connections where the effort variable is the same for all connected components while the flow variable

is conserved. Continuous circuit representation along with switching circuits and switching cells are discussed in [13], [22], [23], and briefly discussed in this paragraph that will focus on switching circuits since continuous circuits are the special case of the more general case switching circuits [24]. Different switching circuits representation techniques were developed, however in this work, switched power junctions technique is to be utilized due to analytical and physical reasons of causality assignment and discontinuities mentioned in [25], [26]. Switching cells are to be modelled as 1 s and 0 s connections governed by its control variable D (analogous to duty cycle in circuits analogy), which links point (C) with Point (A) and (B) via the control variable (D) as shown in Fig. 2. Every SPST is represented as a 1s-junction with two flow determining bonds. The physical realization is complete when the current interruption, when the SPST switch is turned OFF, is depicted as one flow decider bond modelled as a zero value current source (I_s) and the other flow decider bond still linked to the system. Fig. 5 shows the graph representation of three phase inverter circuit with and without filter configurations.

B. GRAPH NEURAL NETWORKS

Graph neural networks (GNNs) are a form of neural networks that operate on graph-structured data, such as social networks, molecular graphs, or knowledge graphs, and consists of three main components: node features, edge features, and graph features as shown in Fig. 1. Node features are the attributes of the nodes in the graph, while edge features are the attributes of the edges in the graph, such as weights or types. Edge indices or adjacency matrix define the nodes that are linked together. Graph features are the global attributes of the graph as a whole, such as performance or density. GNNs learn to update the nodes, edges, and graph features by aggregating the information within the local neighborhood of each node or edge. The output of a GNN can be either the updated node, edge, or graph features, or a prediction based on them after being processed by an additional function. To feed an electric circuit represented as a bond graph to a GNN, the node, edge, and graph features are to be defined, so that GNNs learn updated node, edge, and graph features by applying neural network layers that aggregate information from the local neighborhood of each node or edge. The prediction can be compared with the ground truth values obtained from the circuit simulation or measurement, and the loss function can be used to train the GNN parameters.

C. TRANSFER LEARNING APPLICABILITY

Transfer learning is a technique that leverages the knowledge learned from one domain or task to another domain or task in order to save time and resources from having to train multiple machine learning models from scratch to complete similar tasks. In principle, transfer learning can be applied to any GNN model, yet it is of great importance to apply transfer learning when it is applicable, since not in all cases proves usefulness like:

- When access to new data is not always available or inaccessible[27].
- In the case of a small dataset and the model is a deep neural network, which require large dataset [27].
- When the intended ML task for source and target are similar, the performance of the target network can be improved [28].
- In the case of discrepancy between data domains and distributions [29].

Overall, none of the mentioned cases where transfer learning is useful or encouraged, hence it is concluded that transfer learning is doesn't help with training or generalization in this case.

D. FEATURE ASSIGNMENT PROCESS

The circuit graphs undergo in a feature extraction process so that the final graphs has embedded features like the one shown in Fig 2, which shows a graph representation of single pole double throw (SPDT) switching network. Node features, which are attributes or properties of each node, are to be assigned for every graph node, representing properties such as the type and value of the component. Like wise, edge features represent circuit attributes or properties mapped as graph edge, such as the duty cycle. One-Hot encoding [30] is used to define the node type, which is a way of representing categorical data using binary values, and is used when describing and differentiating between different node types or categories. The node and edge features assignment process can be described as following:

- Nodes representing circuit elements and sources (depicted as blue node in bond graphs), such as resistors, capacitors, inductors, current or voltage sources, the node features are a stack of the one-hot encoding of the node type and the analog value of the element value, indicating the node type and magnitude.
- Same concept is applied to zero current valued current source connected to switched 1-s or 0-s junctions to represent the current interruption in the switch as per bond graph terminology. The node features are combination of the one-hot encoding of the node type plus the analog value of the element value. This analog value can include different controller parameters like the modulation index or frequency, since they can control how the often the current is interrupted in the switched branch by the means of zero current over time. This allows for different modes of operation and controller schemes for the circuit.
- For nodes representing zero or one junctions in bond graph, one-hot encoding of the node ID plus the switch current or voltage value (I_{sw}) obtained from simulation are concatenated all together. Since **switched** zero or one junction are the general cases of ordinary zero or one junction [31], they are assigned the same ID either for zero or for one junction. Other switch parameters or properties like C_{ds} or R_{ds}.. etc can be added to the feature vector.

- Duty cycle, indicating the ratio of the ON to the total time period (T_s), is to be represented as and edge feature with a continuous value from 0 to 1.

It is to be noted that the size of feature matrix is not fixed, and the number of analog features that can be included in the feature matrix are not either. Other values that represent circuit properties that are not mentioned can be also included in the feature matrix. This is indicated by (xx) value in the feature matrix, and it is up to the model designer of how to set these values. However, all feature vectors must have the same size so that the feature matrix is consistent. With the mentioned structure of feature vector, the feature matrix is compatible with the scalability and usability requirements that are discussed later in this paper. This mixture of node and edge feature assignment allows for a more representative features and hence better model performance on the dataset. In the final stage of this work, the graph dataset is to be fed to a graph neural network (GNN) model to obtain circuit predictions for three phase converter circuits.

#	Node		Feature Matrix								
			V	I	L	R	C	1	0	$Val1$	$Val2$	...
1	V_s		1	0	0	0	0	0	0	V_s	xx	xx
2	1		0	0	0	0	0	1	0	Ix	xx	xx
3	0		0	0	0	0	0	0	1	Vs	xx	xx
4	$1s$		0	0	0	0	0	1	0	I_{sw1}	R_{ds}	C_{ds}
5	I_s		0	1	0	0	0	0	0	F	M	xx
6	$0s$	=	0	0	0	0	0	0	1	V_{ph}	xx	xx
7	$1s$		0	0	0	0	0	1	0	I_{sw2}	R_{ds}	C_{ds}
8	I_s		0	1	0	0	0	0	0	F	M	xx
9	0		0	0	0	0	0	0	1	$-V_s$	xx	xx
10	1		0	0	0	0	0	1	0	I_{ph}	xx	xx
11	R		0	0	0	1	0	0	0	R	xx	xx
12	0		0	0	0	0	0	0	1	V_{gnd}	xx	xx

(1)

E. THREE-PHASE INVERTER FEATURE ASSIGNMENT

One leg (phase) of the three-phase inverter bond graph shown in Fig. 3 is used as an example, where the nodes are numbered and the features are assigned accordingly. The node features consisting of the mentioned one-hot encoding and the analog values corresponding to the nature of the element are assigned to every node and is shown in equation (1), where V_{gnd} is the ground voltage, which is equal to zero. Switch characteristics like R_{ds} and C_{ds} representing the drain-source resistance and capacitance respectively are also included as features. I_{sw1} and I_{sw2} are the switch current, V_ph and I_ph are the phase voltage and current, while I_x is the voltage source (V_s) current. F and M are the frequency and the modulation index. By extending this methodology and assigning features to the whole bond graph, a final graph with features annotated can be obtained and is shown in Fig. 4.

IV. MODEL SCALABILITY & USABILITY INVESTIGATION

Scalability is defined as the capability of the prediction model to handle increasing circuit complexity in terms of order

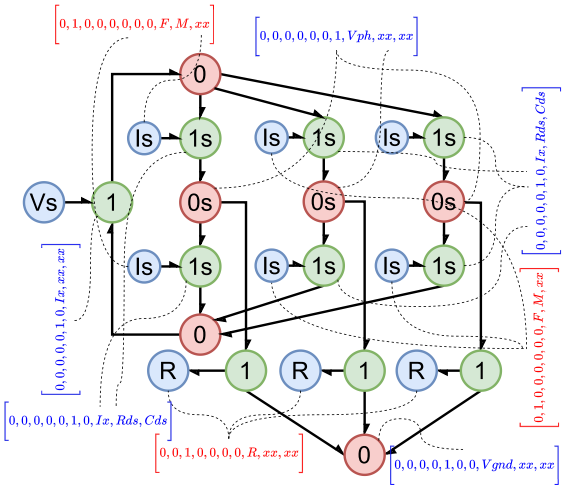

FIGURE 4. Three-phase inverter bond graph in Fig. 5(b) after features have been assigned.

TABLE 1. Dataset Circuit Parameters Range

Parameter	Range	Notes
R	$1\ \Omega \mapsto 100\Omega$	Parameters based on real switches and converter values
F	10 KHz $\mapsto 1MHz$	
R_{ds}	$50\text{m}\ \Omega \mapsto 2\Omega$	
C_{ds}	10 pF $\mapsto 200pF$	
C_f	10nF	
M	$0.05 \mapsto 1$	
L_f	50 µH	

Three Phase Inverter - R Load
Line Voltage prediction error distribution

(number of inductive and capacitive elements) or connection complexity, without introducing model structural changes to adapt input dataset changes. In other words, a scalable model allows for any circuit order, any number of active or passive components as well as any number of connections in circuit to be fed as a dataset to the model. In this paper, scalability is tested by:

- Subjecting the model to complex input circuits like three phase inverter and its variants of even more circuit components (and hence more nodes) while the model structure is kept fixed. The aim is to test the model's response to an increasingly large circuit dataset to determine how well it can perform on a larger input circuit size.
- Scaling up the complexity of prediction by increasing number of predicted variables obtained from model output from single variable to multi-variable regression problem.

Additionally, the model is tested for **usability**, which is defined as a measure of how flexible the model is for user input/parameter changes, including adaptations to hardware/control variables changes by accommodating the growth of data and input size. A usable model offers an interactive,

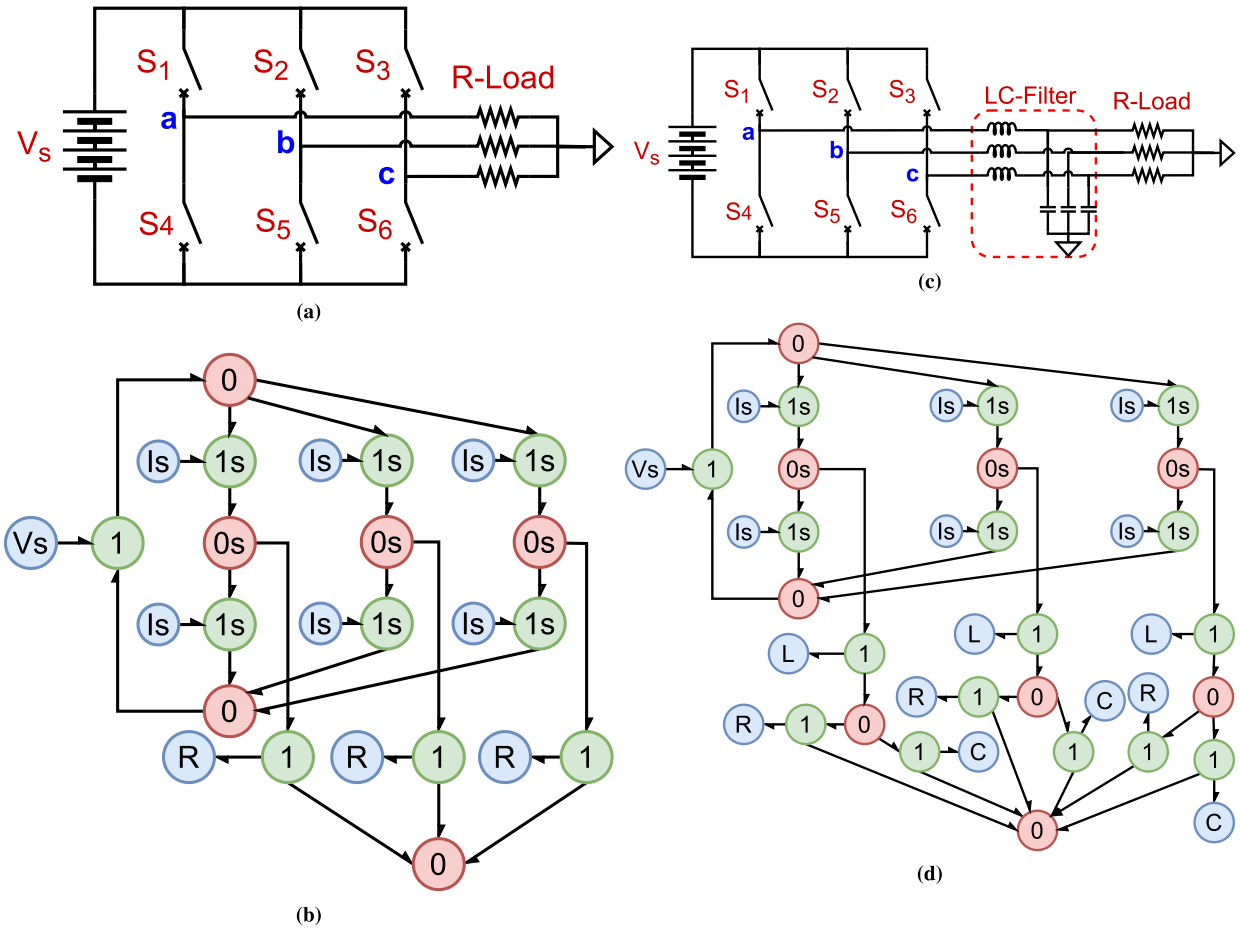

FIGURE 5. Three phase DC-AC inverter circuit with R-load and its equivalent graph: (a) & (b) No Filter, (c) & (d) with LC-filter.

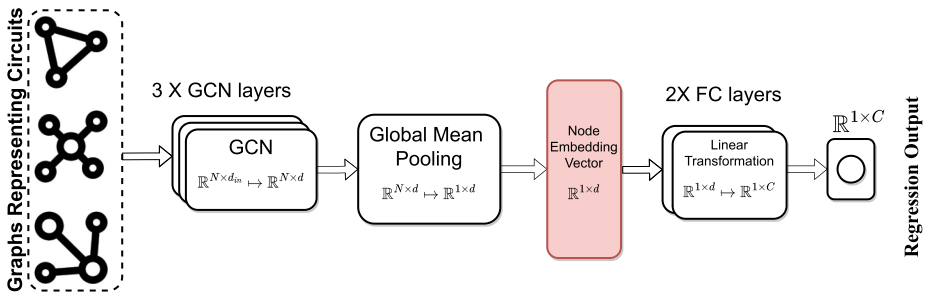

FIGURE 6. Regression model structure.

easy to use and intuitive technique for feeding circuit as input and offers a standard for manipulating hardware changes and control parameters/schemes changes. The paper shows model's usability by demonstrating the model prediction capability when

- Load (R), source (V_{in}) and switching frequency (F) changes.
- Parasitic components variation like R_{ds}, C_{ds}.
- Different control parameter are varied as well as model response to control scheme variation.

Lastly, the model's ability to accurately predict is a main concern when assessing a regression model. The model's predictive capabilities are tested by comparing its predictions to known results obtained from simulations and analyzing residuals. The paper analyse the model performance according to multiple aspects:

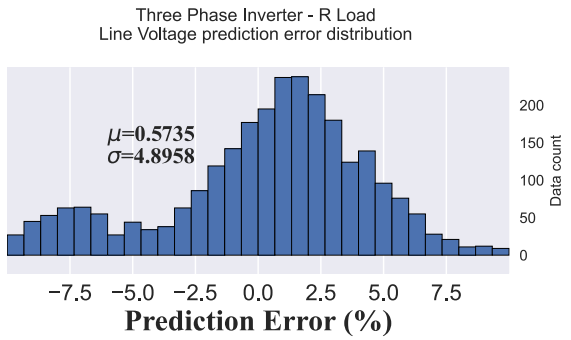

FIGURE 7. Model prediction percentage error when the inverter is feeding an R-load.

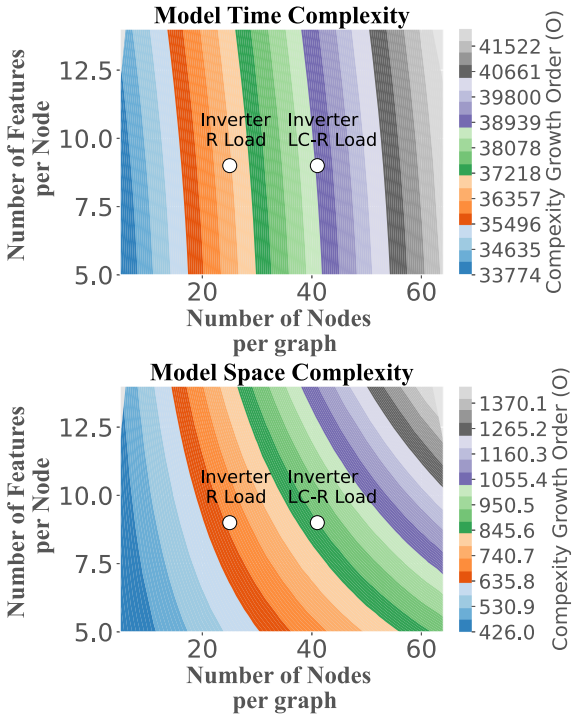

FIGURE 8. Time and space complexity of the proposed model.

- Comparing recorded simulation results to predictions, allowing for a mathematical analysis of how well the model can predict a known outcome.
- Analyzing residuals allows for investigating the difference between the predicted and the simulated outcomes.
- Provide histograms about the error distribution along different prediction variables and versus parameter variations, which helps to identify if the model is failing to account for parameter variations.

A. SCALABILITY AND USABILITY OF BOND GRAPHS

Bond graph modelling achieves **scalability** and **usability** by combining graphical, modular, hierarchical, and dynamical modelling properties to model complex physical dynamic systems in a unified methodology, which is explained as follows:

- BG can handle multiple energy domains (mechanical, electrical, thermal and chemical) seamlessly and consistently, using the same set of elements and variables.
- BG utilizes the concept of power bonds to connect different elements and represent the bi-directional exchange of energy between them, which abide to the conservation of mass and energy laws in the system.
- BG allows the integration of thermodynamics to incorporate the effects of entropy, heat transfer, and chemical reactions in any physical system, enabling the modelling of non-linear and irreversible phenomena in power systems.
- BG inherently supports modularity, allowing for system decomposition/recombined into/from smaller subsystems that are modelled independently, which eventually facilitates the integration of multi-physical domain models.

B. GRAPH NEURAL NETWORK SCALABILITY

In a wide range of applications, Graph Neural Networks (GNN) are a powerful tool for representing non-Euclidean data. The complex structural links seen in graphs are captured by GNNs and are further processed to apply a trained task. They may thus be used to classify whole networks in a single step. GNNs can be trained to predict trends from the global structure, or perform classification tasks across whole networks in a single step. There are many variants of GNNs like GCN [32], GAT [33]), GraphSAGE [34], R-GCN [35]). GCNs have capability to capture complex relationships between nodes in a graph and achieve better prediction accuracy [32], generalize to unseen data, and accommodate nodes of varying degree of connectivity. GNNs can achieve scalability when circuits are fed as graphs by inherently applying the following techniques:

- *Message passing:* which allows GNNs to propagate information between nodes and edges, where each node updates its representation by aggregating messages from self to neighbors, allowing for capturing self and neighborhood information and eventually overall global structure of the graph.
- *Permutation invariant functions:* where the order of the nodes in the graph is not of importance and doesn't affect the operations performed on graph input, allowing for handling of graphs of different sizes and orders without changing graph architecture or parameters.
- *Attention mechanisms:* which assigns different weights to the messages from different node neighbors, based on relevance and importance. This can help GNNs focus on the most informative parts of the graph, and reduce the noise and redundancy from less relevant connections.

V. REGRESSION MODEL FOR CONVERTER DYNAMICS PREDICTION

This section presents a proposed GNN based model applied to DC-AC inverter circuits in order to obtain predictions based on circuit topology and component values. Multiple

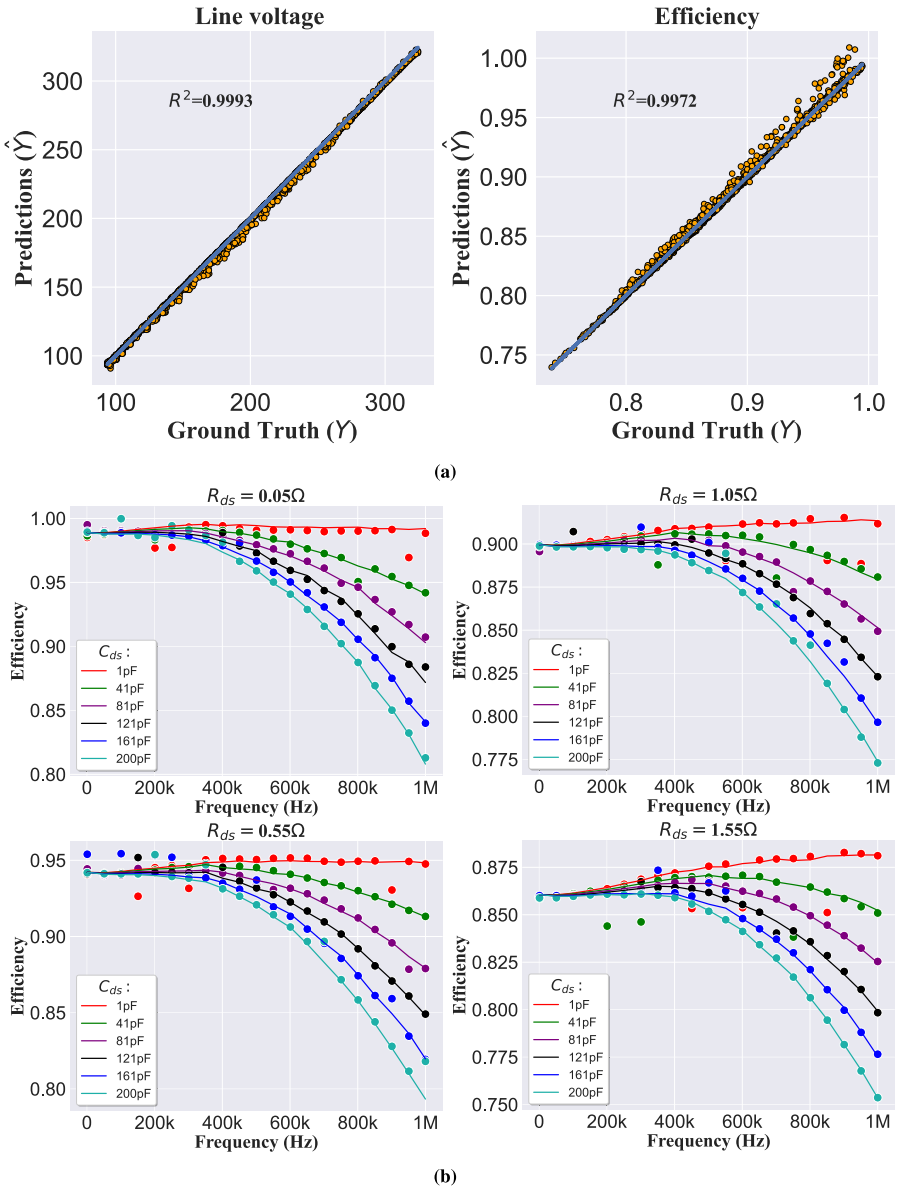

FIGURE 9. (a) Actual vs prediction in line voltage and efficiency outputs, (b) Efficiency prediction when parasitic resistance and capacitance vs the change along the switching frequency range.

case studies including single and multi-variable regression problems are shown, including obtaining predictions of the most essential outputs of any converter, namely **line voltage and efficiency**, with the potential to scale up to include many more variables. The model is to be fed a dataset that contains circuit data after being transformed into its graph forms and assigned node and edge features, as well as information about the prediction targets, obtained from simulations.

A. REGRESSION MODEL STRUCTURE

The neural network model takes converter circuits in graph forms (G), node features (X) expressing element type and element value, adjacency matrix (A), edge features (Z) as input, and outputs the predicted variables (Y) with output vector size being the number of predicted variables (C). The mathematical representation of the regression model and the propagation of graph features across layers are given by (2)–(7).

$$Y = Regression(X, A, Z) \tag{2}$$

Where

$$X \in \mathbb{R}^{N \times d_{in}} \tag{3}$$

$$Y \in \mathbb{R}^{C \times 1} \tag{4}$$

$$GCN^{(k)} : \mathbb{R}^{N \times d_{in}} \mapsto \mathbb{R}^{N \times d} \tag{5}$$

$$k \in \{0, 1, .., k-1\}$$

$$GMR : \mathbb{R}^{N \times d} \mapsto \mathbb{R}^{1 \times d} \tag{6}$$

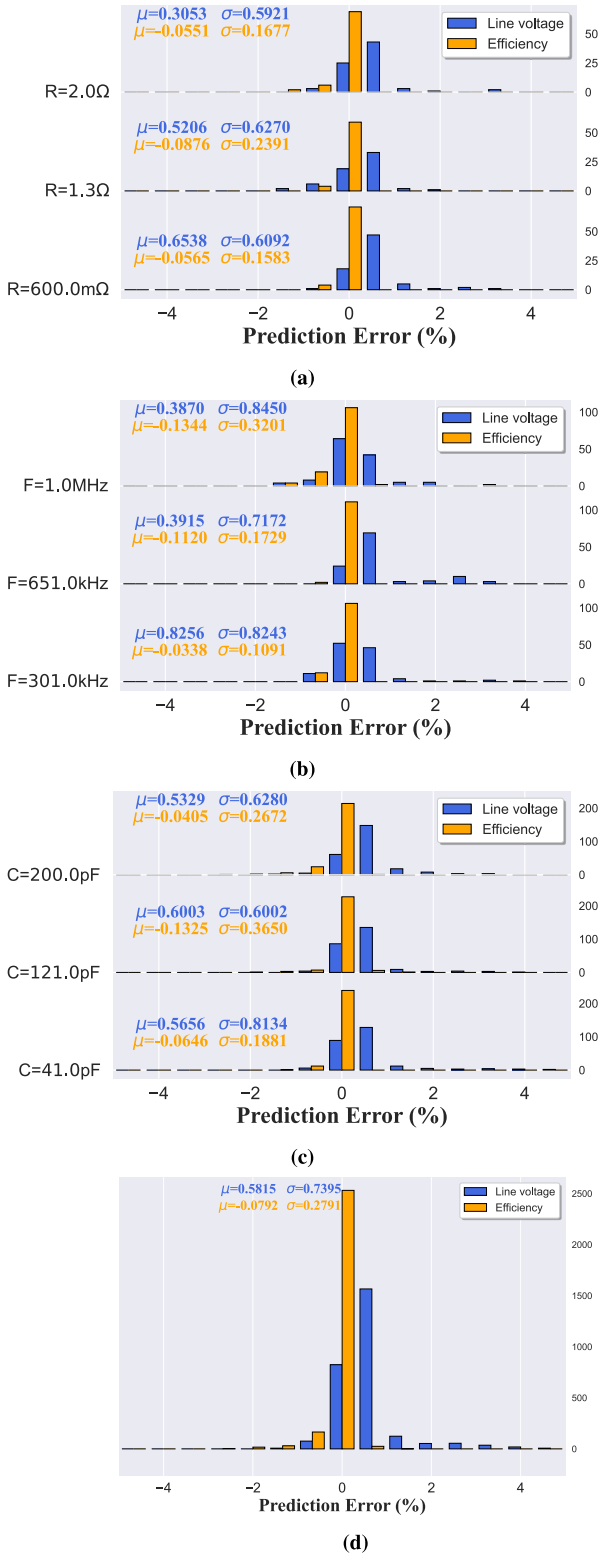

FIGURE 10. Model percentage prediction error distributions when subjected to: (a) Resistance variations, (b) Frequency variations, (c) Capacitance variations, while (d) is the overall model response to variations.

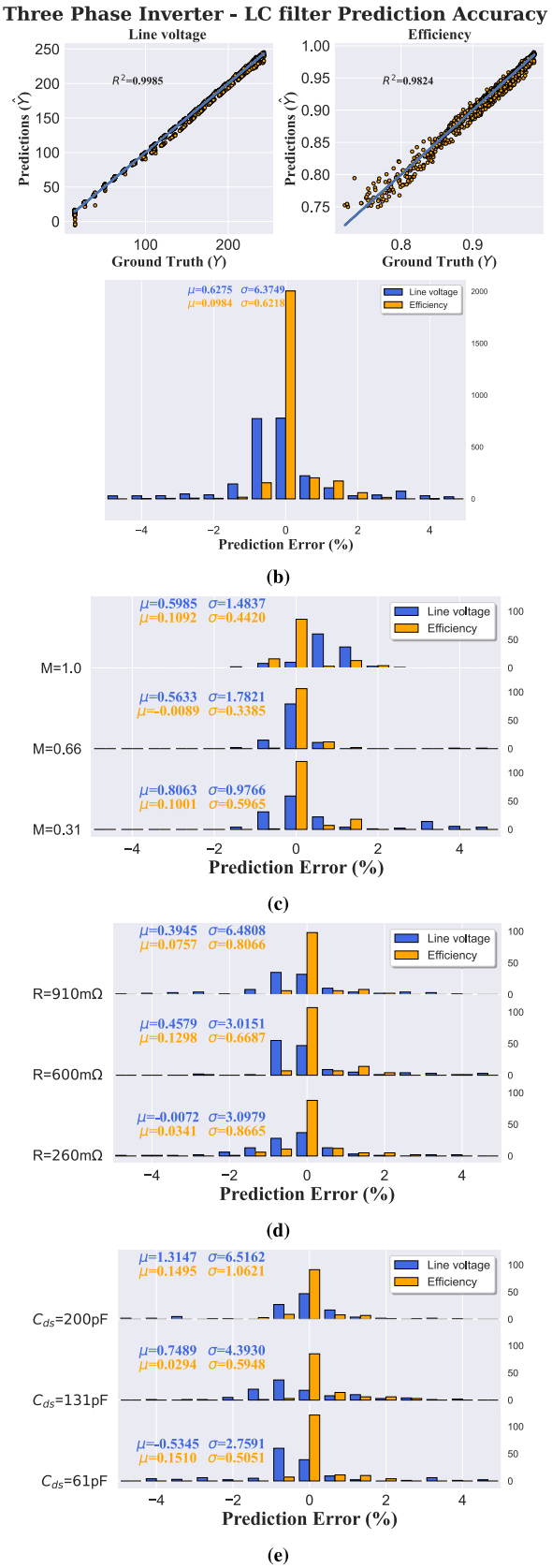

FIGURE 11. Multi-variable SPWM controlled Inverter prediction error (%).

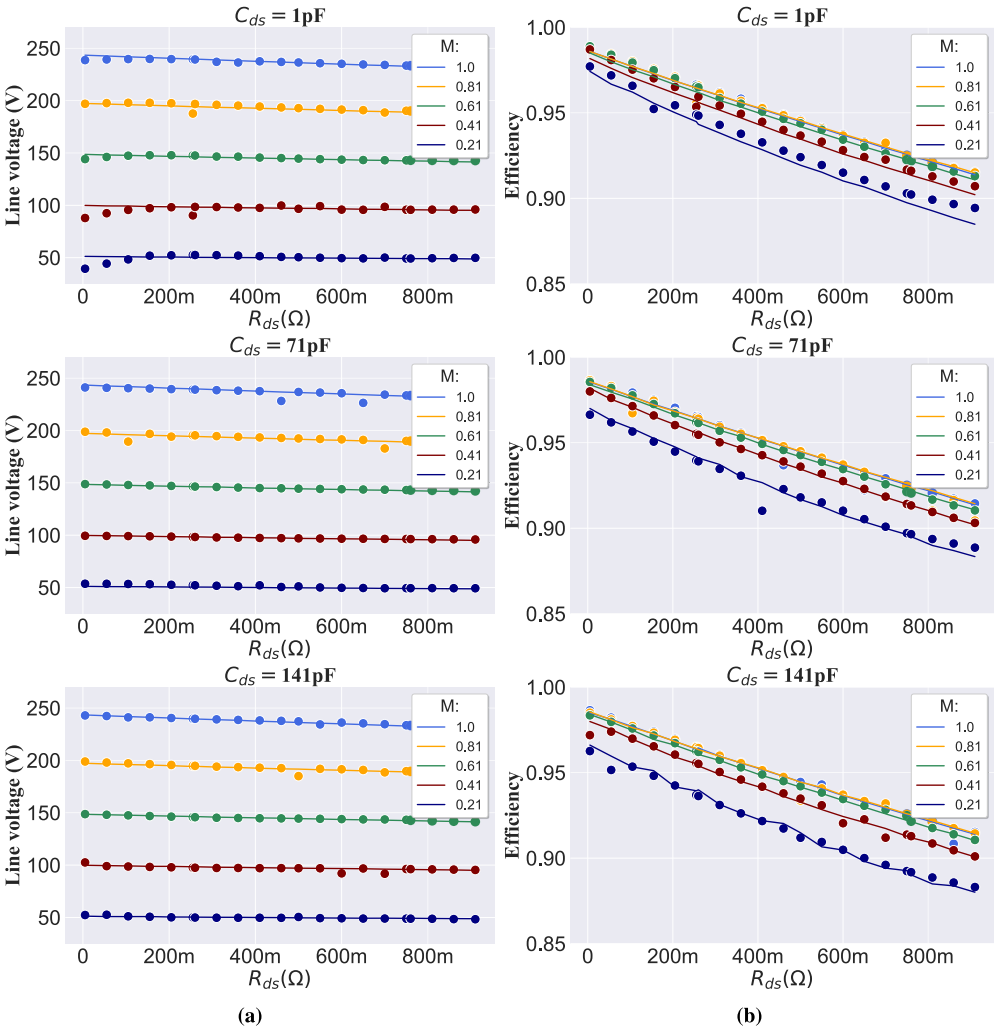

FIGURE 12. **Multi-variable prediction output for DC-AC inverter with LC filter: (a) Line voltage prediction, (b) efficiency prediction, where *M* is the modulation index.**

$$FC : \mathbb{R}^{1\times d} \mapsto \mathbb{R}^{1\times C} \tag{7}$$

Mathematically, this initial embedding function is represented by (8). The aggregation layer has multiple Graph Convolution Networks (GCN) that performs multiple message passing leaps to collect information about neighbouring nodes and keeps updating the latent dimensional vector with dimension d, which is mathematically represented as in (9).

$$X^{(0)} = E(X) \tag{8}$$

$$X^{(k+1)} = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}X^{k}\Theta^{k}\right) \tag{9}$$

$$\mathbf{x}_i' = \mathbf{\Theta}^{\top} \sum_{j\in\mathcal{N}(v)\cup\{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j\hat{d}_i}}\mathbf{x}_j \tag{10}$$

where Θ^k is a weight matrix for the k-th neural network layer and σ is a non-linear activation function like the rectified linear unit (ReLU), $\hat{A}$= A + I, where I is the identity matrix and $\hat{D}$ is the diagonal node degree matrix of $\hat{A}$. This allows the GCN to scale well, because the number of parameters in the model is not tied to the size of the graph. The node-wise formulation of feature update is given by (10), where $\hat{d}_i = 1 + \sum_{j\in\mathcal{N}(i)} e_{j,i}$ denotes the edge weight $e_{j,i}$ from source node j to target node i.

Fig. 6 shows a block diagram of obtaining predictions from circuits using a regression model. The model utilizes three GCN layers to exchange messages across nodes. The output is fed to the global mean readout (GMR) layer, which averages the processed node and edge features to an output dimension of d. The two fully connected (FC) linear layer is trained to linearly transform the averaged graph vector to desired output predictions by minimizing the mean square error loss function. According to the universal approximation theorem, FC layers may estimate any function without any limitation on the structure [36]. Equations (1)–(6) express the regression model mathematics, while (11)–(12) are used to express the two-layer FC layer mathematically, where w_0 & w_n are the bias and weight, respectively, while x is the input to the linear

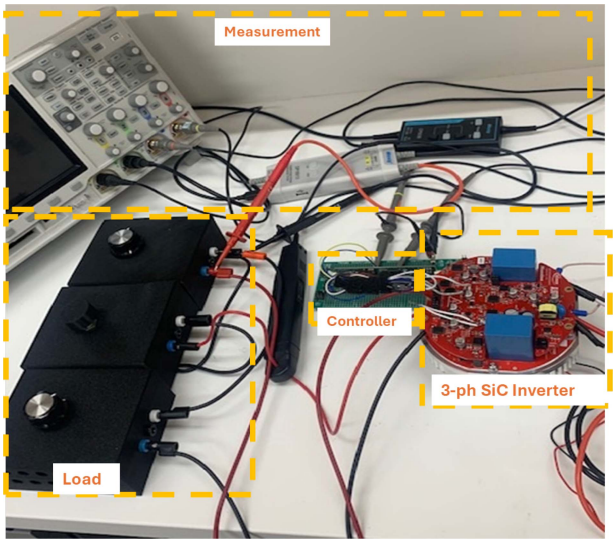

FIGURE 13. Experimental setup used to record experimental data.

layer, N is the number of neurons, and *sigma* is the activation function.

$$f^{(1)}(\mathbf{x}) = \sigma\left(w_0^{(1)} + \sum_{n=1}^{N} w_n^{(1)} x_n\right) \tag{11}$$

$$f^{(2)}\left(f^{(1)}(\mathbf{x})\right) = w_0^{(2)} + \sum_{i=1}^{U_1} w_i^{(2)} f_i^{(1)}(\mathbf{x}) \tag{12}$$

$$f(\mathbf{x}) = \max(\alpha\mathbf{x}, \mathbf{x}) \tag{13}$$

B. DYNAMICS PREDICTION CASE STUDIES

To demonstrate the model's ability to handle different converter circuits, the paper presents multiple case studies with various circuit parameters, including hardware components (such as an LC filter), controller schemes (such as square and sinusoidal PWM), and various system variables variations (such as load, frequency, modulation index and input source). The paper conducts experiments on a DC-AC inverter with different combinations of these parameters, and observes how the model performs under these changes. The model training uses a dataset of circuit graphs with different parameters and their true outputs, and tries to fit the output without memorizing the data (overfitting).

C. DATASET DATA RANGE

The dataset range must include the control and circuit parameter changes in an inclusive and expressive manner for an accurate output estimation by the proposed regression model. Generally, selected algorithms should have the potential to accommodate minimum and maximum values in the datasets. Additionally, the range of the datasets should effectively match the purpose of the machine learning model. If the dataset ranges are too narrow, the model may only be able to react to changes in parameters and will not be able to predict the output correctly, which may become a limiting factor. On the other hand, if the data ranges are too wide, the model may become too generalized, and its accuracy may suffer due to incorrect predictions. The range of the utilized datasets in all cases studies including circuit and control parameters represented as node and edge features are listed in Table 1. The variables ranges [resistance (R), inductance (L), voltage(V), Parasitic resistance (R_{ds}), inductance (L_s) and capacitance (C_{ds}) and frequency (F)] are selected based on true values obtained from datasheets of several circuit components.

VI. RESULTS ANALYSIS

Histograms shown in Fig. 7 highlights the prediction error across multiple prediction scenarios when inverter connected to resistive load with and without LC filter. The histograms also include the scenario where multiple prediction output is required. It is shown that the model can attain high prediction accuracy, while maintaining prediction error percentage less than 10%.

A. PREDICTION ACCURACY EVALUATION

The coefficient of determination (R-squared) is a statistical measure of how well a model captures the variability of a dependent variable, expressed as the percentage of variation explained by the model. The (14) shows the formula for the coefficient of determination, with RSS is the sum of squares of residuals and TSS is the total sum of squares. In general, higher R-squared values denote a better fit of the model to the observed data, indicating smaller differences between the true values and the predicted values. The R-squared score recorded by proposed model when applied to the testing dataset is 99.49%. Moreover, in Fig. 9(a), the ground truth values obtained from simulation (solid line) and the predicted values for selected prediction targets (line voltage and converter efficiency) are shown. The model exhibits high stability and adaptability in terms of its complex predictions, responding to variations in frequency, drain-source resistance and capacitance, which are the main factors affecting both outputs. Detailed model analytics and in depth circuit variable- prediction accuracy analysis is given.

$$R^2 = 1 - \frac{R_{SS}}{T_{SS}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2} \tag{14}$$

where, y_i = observed value, $\hat{y}_i$ = predicted value, $\bar{y}_i$ = mean of observed values.

B. CIRCUIT COMPLEXITY INVESTIGATION

In bond graph notation, when the circuit complexity is increased, i.e more components are added or a more complex circuit is considered, the number of connection junctions as well as the number of component nodes consequently change. The same concept applies to GCNs and hence the prediction model can accommodate more complex circuit representations. This is shown when the model accommodated the three phase DC-AC inverter circuit and its including the parameters

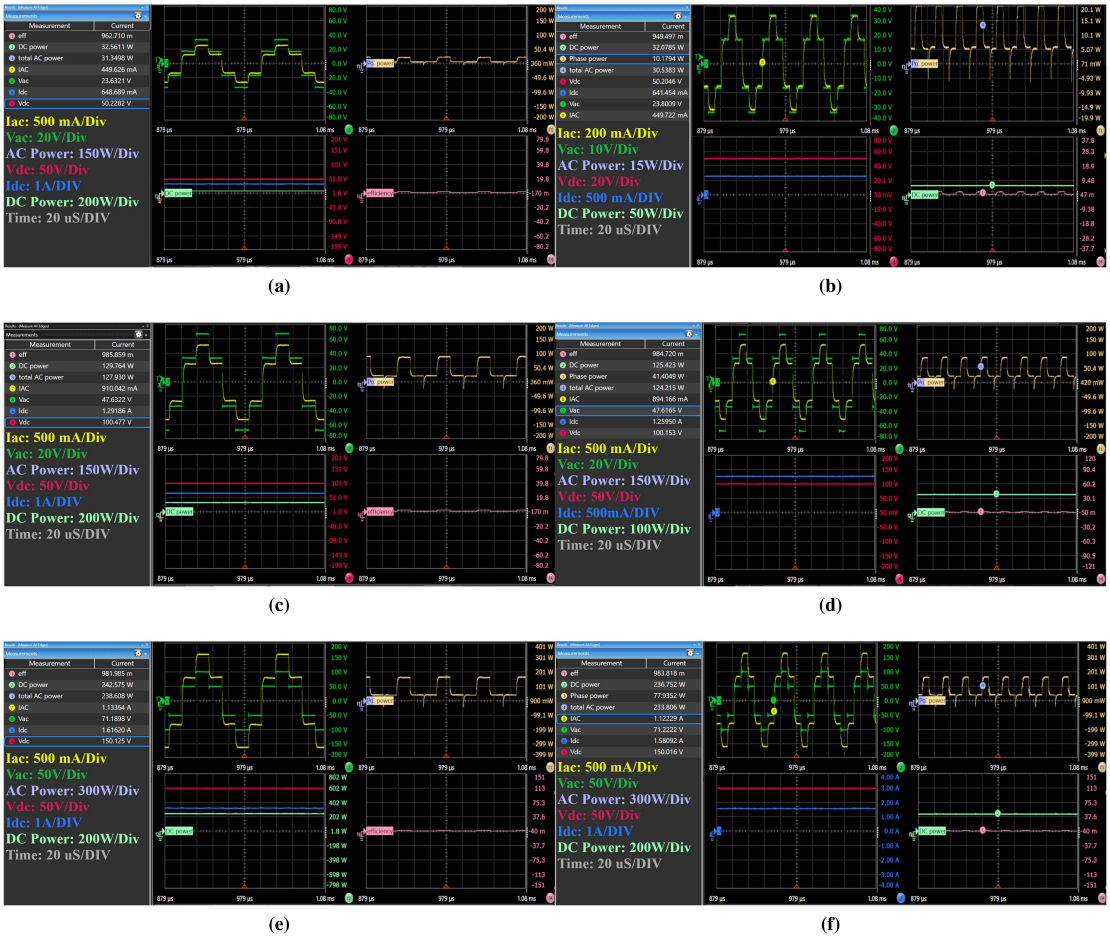

FIGURE 14. Scope measurements samples of DC-AC inverter operating under different loads,frequencies and supply voltages: (a) 50 V -50Ω - 10 Khz, (b) 50 V -50Ω - 20 Khz, (c) 100 V -50Ω - 10 Khz, (d) 100 V -50Ω - 20 Khz, (e) 150 V -60Ω - 10 Khz, (f) 150 V -60Ω - 20 Khz.

and the load variations. In order to To quantify the model ability to handle more prediction outputs and more complex circuits, model performance metrics are to be used as judgement factors. Some possible metrics are:

- *Accuracy:* Evaluation of the GNN model prediction percentage, by using one of the most common evaluation metrics like F1-score, accuracy, precision or R^2.
- *Computational scalability:* Computational effort evaluation of model's performance,number of parameters or memory usage against the increase in graph nodes and feature sizes without compromising the accuracy or performance.

Given G as the circuit graph, E as the number of edges, N as the number of nodes, and F as the feature vector length, the computational effort can be broken down to time and space complexities and are calculated as:

- Time complexity: $O(3(E + NF^2) + (N + 128N) + 128 \times 128 + 128 \times 128 + 128 \times 2)$
- Space complexity: $O(N + E + NF + 128 + 128 + 128 + 2)$

Fig. 8 shows the time and space complexity of the proposed model. (O) is the order of magnitude which defines the complexity growth proportional to the graph input size and number of features assigned for every node. A comparison table shown in Table 2 shows the percentage change in fitting accuracy across the testing cases the model has been through.

TABLE 2. Predication Model Performance Across Multiple Case Studies

Circuit	Case	Line Voltage	Efficiency
R-Load	**Single-variable Regression** **Square Wave Modulation** -Sweep variables: Vin, Rload, F	μ= 0.5735% σ= 4.89%	N/A
R-Load with LC filter	**Single-variable Regression** **Square Wave Modulation** -LC filter added -Sweep variables: Vin, Rload, F	μ= 1.2405% σ= 12.8741%	N/A
R-Load with LC filter	**Multi-variable Regression** **Square Wave Modulation** -Sweep variables: R_{ds}, C_{ds}, F	μ=0.5815% σ=0.7395%	μ=0.0792% σ= 0.2791%
R-Load with LC filter	**Multi-variable Regression** **Sine Wave Modulation** -Sweep variables: R_{ds}, C_{ds}, M	μ=0.2675% σ=6.3749%	μ=0.0984% σ= 0.6218%

The prediction error mean and standard deviation are crucial for its performance as these defining metrics highlight the accuracy of the model's predictions. A low error mean (μ) and standard deviation (σ) typically indicate a high-performing model, whereas a high error μ and σ indicate a

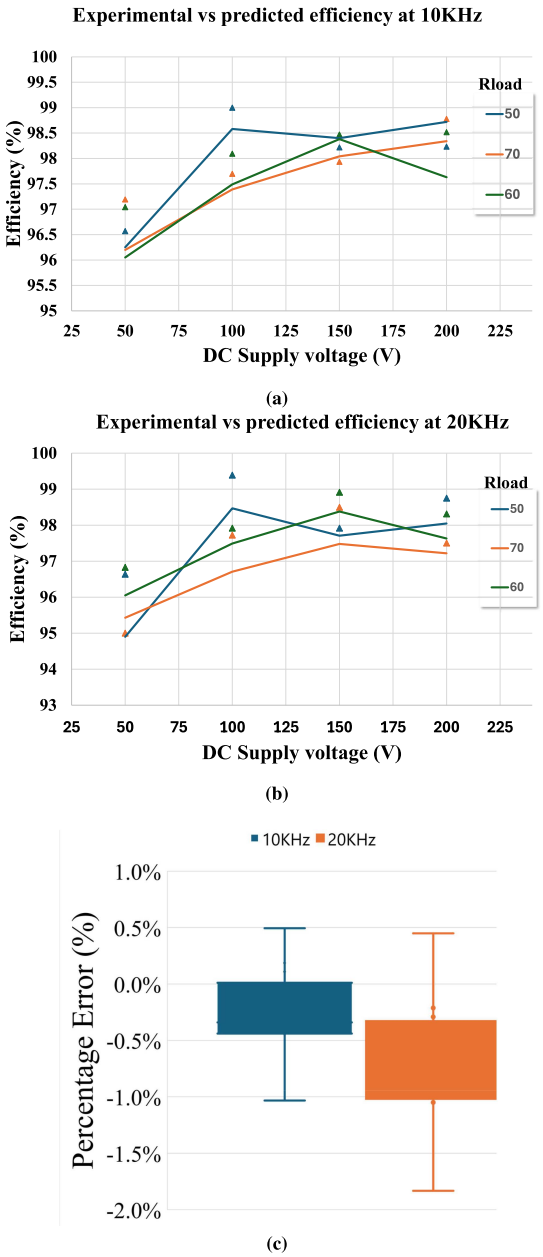

FIGURE 15. Experimental Vs predicted converter efficiency at different frequencies: (a) 10 KHz, (b) 20 Khz. (c) Model prediction Percentage error visualization. —: Practical efficiency △: Predicted efficiency.

low-performing model. Keeping both performance metrics in low values is necessary for optimal performance.

C. CIRCUIT & CONTROL PARAMETERS SENSITIVITY ANALYSIS

Model performance when subjected to parameter variations is analysed to determine the model accuracy and error response to parameter values, and quantify their effect the results. The primary goal of this analysis is to assess how the model accounts for multiple circuit parameter variations as well as control parameter variations and their effect on the output. Fig. 9(a) shows the ground truth values (Y) obtained from simulations versus predicted ($\hat{Y}$) outputs of the model, namely, line voltage and efficiency. The straight line represents the ideal case of the model having 100% accuracy, while the dots represents the predicted output at this instant. R^2 score value of 99.93% for predicting line voltage and 99.72% for predicting efficiency was validated for the model output. Moreover, Fig 9(b) presents the individual visualization of each input variable vs the model prediction and its effect on the multi-variable regression output. Histograms of the prediction error percentage distribution across the multi-variable regression model including mean and standard deviation for each circuit parameter are discussed in details in the next subsections.

1) CIRCUIT PARAMETERS VARIATION (HARDWARE VARIATION)

Two types of variations are tested with the proposed regression model, namely hardware and controller parameter variations. Hardware variation are when circuit component values are changed, which is a real life equivalent of changing resistor values of replacing a mosfet with a lower parasitic one. The histogram in Fig. 10(a) indicates error distribution in efficiency prediction when exposed to changes in load resistance across dataset range. The overall performance of the model was highly accurate, as indicated by the minimal prediction error. Furthermore, the error illustrated in Fig. 10(c) is minor when parasitic capacitance is changed. Although the values ranges of the predicted outputs are different and vary significantly, the model was able to accurately obtain prediction with less than ±2 % error shown in the histograms across all parasitic resistance and capacitance range.

2) CONTROL PARAMETERS VARIATIONS

Controller parameters are changed like modulation scheme and frequency, which in real world applications are controlled by digital controllers running in real-time and require software changes. The proposed regression model can accurately predict the converter behaviour under these changes.

a) Switching Frequency Variation: Frequency is an important factor for converter operation mode, which is actively managed by a digital controller in real-time, depicted as a node feature when represented in the circuit graph. In Fig. 10(b), the variance in efficiency prediction error is relatively low, with almost no changes across frequency range. On the other hand, when looking into line voltage prediction, the same pattern is observed, yet the prediction errors are comparatively higher, with a higher variance, due to the higher output range.

b) Modulation Scheme Variation: Sinusoidal Pulse Width Modulation (SPWM) is widely used to control the output voltage and frequency of DC-AC inverters, which involves modulated pulses generated according to a sinusoidal reference signal with predetermined amplitude and frequency. Fig. 11 shows the filtered (using LC filter) inverter output predictions and the corresponding prediction error, which is

TABLE 3. Experimental Setup Components

Component	Description
DC source	50V → 200V
DC-AC Converter: REF-DR3KIMBGSICMA	3 phase DC-AC inverter Input voltage 350→ 800 VDC Output voltage 220 → 480VAC
Mosfet switch: IMBG120R045M1H	45 m Ω @ 25 °C 1200V 47A @ 25 °C
Output inductance	40 µH
Load resistance	50 → 70 Ω
Frequency	10KHz → 20KHz

relatively low across variable variations of drain source resistance and capacitance (R_{ds}, C_{ds}) and modulation index (M). Fig. 12 shows the predicted line voltage and efficiency across mentioned variables. Despite the irregular slope change between predicting the line voltage and the efficiency, the model was able to adapt to variations and obtain minimal error predictions.

VII. EXPERIMENTAL VERIFICATION

In this section, the accuracy of the proposed GNN model and graph framework for predicting the performance of converter circuits based on their topology and parameters is validated.

A. EXPERIMENTAL SETUP AND METHODOLOGY

A three-phase DC-AC inverter circuit identical to the case study in Fig. 5(a) is used for verification, while hardware and software variations are applied to the system. The converter circuit exhibits two system variation, namely hardware variations like load resistance variations and source voltage level variations, and control variations like switching frequency variations. The components of the experimental setup as well as their values and variation ranges are listed in Table 3. Two data sources are used for the experiments: simulated data generated from simulation model that emulates the exact practical and environmental circumstances, and measured data obtained from practical experiments at different converter operating points.

1) SIMULATED DATA

The simulated data are generated by using the published LTspice model available in [37] running on LTspice circuit simulator, allowing for the capturing of converter circuits behavior and response at different scenarios and splitting the data into training and test sets by 70% to 20% ratio respectively. The graph representations of the converter circuits are constructed by using the bond graph modelling technique, as described in Section III. Node and edge features are assigned to represent the circuit elements and parameters, while the same graph structure is used for different scenarios and parameter variations in the experiment.

2) MEASURED DATA

The measured data are obtained by using a hardware setup consisting of a three-phase DC-AC inverter connected to an R-load as shown in Fig. 13. Data are collected individually and independently of other variables, i.e earch variable is varied separately while other variables are kept constant. Fig. 14 shows sample points of the recorded output data of the DC-AC inverter operation under variable input voltage, load resistance and frequency.

B. EXPERIMENTAL RESULTS AND ANALYSIS

In this case study, the GNN model is tested on predicting the line voltage and efficiency of a three-phase DC-AC inverter connected to an R-load, as a function of a single input variable. The predictions of the GNN model are compared with the ground truth experimental values. The predictions and the ground truth values are plotted in Fig. 15(a) and (b), which shows the model predictions at 10 and 20 KHz frequencies. Fig. 15(c) shows the overall percentage error of the model expressed as percentage error at two different operating frequencies, indicating that the model can explain most of the variance in the data. Generally, the model is able to predict the converter efficiency within +0.5% to -2%, and to extract the converter behavior and efficiency based on the variation of hardware components and controller signals.

VIII. CONCLUSION AND FUTURE RESEARCH

The manuscript showed predictions based on three phase DC-AC inverter under multiple operation scenarios, control signal and component values in multiple study cases for the purpose of assessment. By representing the converter circuit as graphs and applying GNNs, the regression model was able predict circuit performance information as well as identify the type of circuit. The model has proved its ability to be utilized in any circuit configuration or connection and its ability to include any number of circuit elements including passive and active ones. The motioned tests were conducted at different control parameters as well as different circuit components and connections, and verified experimentally against those variations. The proposed GNN model as well as the graph framework were built with generalization and scalability in mind, making it an excellent candidate for solving multiple circuit analysis, control and design problems as follows:

- *Converter design optimization:* Proposed GNN model can be used to optimize the converter design parameters, such as component values, input source, output load, control parameters, and schemes.
- *Converter state detection and diagnosis:* Component failures, short circuits, open circuits, CCM and DCM converter mode,..., etc can be predicted and identified based on the converter performance.
- *Instant simulator:* Proposed model can be used to simulate the converter performance and dynamics instantly once trained, without requiring any circuit simulation software.

- *Real-time parameter estimation and monitoring:* Proposed model can be further reduced and optimized to work on microcontrollers as in [38].
- *AI generated circuits:* Proposed graph framework can be used to generate new and novel converter circuits for domain specific applications based on generative artificial networks with the help of generative AI models.

REFERENCES

[1] M. Xia, H. Shao, X. Ma, and C. W. De Silva, "A stacked GRU-RNN-based approach for predicting renewable energy and electricity load for smart grid operation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 7050–7059, Oct. 2021.

[2] L. Ren, J. Dong, X. Wang, Z. Meng, L. Zhao, and M. J. Deen, "A data-driven auto-CNN-LSTM prediction model for lithium-ion battery remaining useful life," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3478–3487, May 2021.

[3] Y. Ma, Z. He, W. Li, L. Zhang, and B. Yu, *Understanding Graphs in EDA: From Shallow to Deep Learn.*. New York, NY, USA: Association for computing machinery, 2020, pp. 119–126. [Online]. Available: https://doi.org/10.1145/3372780.3378173

[4] H. Wang, J. Yang, H.-S. Lee, and S. Han, "Learning to design circuits," in *Proc. NeurIPS SysML*, 2018, pp. 1–7.

[5] H. Wang et al., "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. 57th ACM/IEEE Des. Automat. Conf. (DAC)*, 2020, pp. 1–6.

[6] H. He and G. Zhang, "End-to-end learning for distributed circuit design," in *Proc. 32nd Conf. Neural Inf. Process. Syst.*, 2018, pp. 1–8.

[7] G. Zhang, H. He, and D. Katabi, "Circuit-GNN: Graph neural networks for distributed circuit design," in *Proc. 36th Int. Conf. Mach. Learn.*, 2019, vol. 97, pp. 7364–7373. [Online]. Available: https://proceedings.mlr.press/v97/zhang19e.html

[8] K. Hakhamaneshi, M. Nassar, M. Phielipp, P. Abbeel, and V. Stojanović, "Pretraining graph neural networks for few-shot analog circuit modeling and design," 2022. [Online]. Available: https://arxiv.org/abs/2203.15913

[9] X. Gao, C. Deng, M. Liu, Z. Zhang, D. Z. Pan, and Y. Lin, *Layout Symmetry Annotation for Analog Circuits With Graph Neural Networks*, New York, NY, USA: Association for computing machinery, 2021, pp. 152–157. [Online]. Available: https://doi.org/10.1145/3394885.3431545

[10] H. Chen, K. Zhu, M. Liu, X. Tang, N. Sun, and D. Z. Pan, "Universal symmetry constraint extraction for analog and mixed-signal circuits with graph neural networks," in *Proc. ACM/IEEE 58th Des. Automat. Conf.*, 2021, pp. 1243–1248.

[11] H. Wang et al., "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. ACM/IEEE 57th Des. Automat. Conf.*, 2020, pp. 1–6.

[12] H. Wang, J. Yang, H. Lee, and S. Han, "Learning to design circuits," 2018, *arXiv:1812.02734*. [Online]. Available: http://arxiv.org/abs/1812.02734

[13] 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.

[14] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to simulate complex physics with graph networks," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 8459–8468.

[15] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia, "Learning mesh-based simulation with graph networks," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–8.

[16] M. Cranmer et al., "Discovering symbolic models from deep learning with inductive biases," in *Proc Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 17429–17442.

[17] M. Lino, C. Cantwell, A. A. Bharath, and S. Fotiadis, "Simulating continuum mechanics with multi-scale graph neural networks," 2021, *arXiv:2106.04900*.

[18] P. J. Gawthrop and G. P. Bevan, "Bond-graph modeling," *IEEE Control Syst. Mag.*, vol. 27, no. 2, pp. 24–45, Apr. 2007.

[19] P. Gawthrop and E. J. Crampin, "Bond graph representation of chemical reaction networks," *IEEE Trans. Nanobiosci.*, vol. 17, no. 4, pp. 449–455, Oct. 2018.

[20] K. Sirivadhna, E. F. Richards, and M. D. Anderson, "The application of bond graphs to electrical machinery and power engineering," *IEEE Power Eng. Rev.*, vol. PER-3, no. 5, pp. 35–35, May 1983.

[21] Y. Baqqal and M. E. hammoumi, "A generic approach of modelling perturbation of mechanical systems: Concept with bond graph," in *Proc. IEEE Int. Conf. Technol. Manage., Operations Decis.*, 2018, pp. 219–223.

[22] A. K. Khamis and M. Agamy, "Mapping continuous circuit structures to machine learning space," in *Proc. IEEE 31st Int. Symp. Ind. Electron.*, 2022, pp. 149–155.

[23] A. K. Khamis and M. Agamy, "Converter circuits to machine learning: Optimal feature selection," in *Proc. IEEE Energy Convers. Congr. Expo.*, 2022, pp. 1–7.

[24] A. Umarikar and L. Umanand, "Modelling of switched mode power converters using bond graph," *IEE Proc. Elect. Power Appl.*, vol. 152, pp. 51–60, 2005.

[25] V. D. Gebben, "Bond Graph Bibliography for 1961–1976," *J. Dyn. Syst., Meas., Control*, vol. 99, no. 2, pp. 143–145, 1977. [Online]. Available: https://doi.org/10.1115/1.3427087

[26] A. Markakis, W. Holderbaum, and B. Potter, "A comparison between bond graphs switching modelling techniques implemented on a boost dc-dc converter," in *Proc. IEEE 33rd Int. Telecommun. Energy Conf.*, 2011, pp. 1–7.

[27] A. Farahani, B. Pourshojae, K. Rasheed, and H. R. Arabnia, "A concise review of transfer learning," in *Proc. IEEE Int. Conf. Comput. Sci. Comput. Intell.*, 2020, pp. 344–351.

[28] W. Zhang, Y. Fang, and Z. Ma, "The effect of task similarity on deep transfer learning," in *Proc. Int. Conf. Neural Inf. Process*, 2017, Art. no. 31151731. [Online]. Available: https://api.semanticscholar.org/CorpusID

[29] F. Zhuang et al., "A comprehensive survey on transfer learning," *Proc. IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021.

[30] F. Harrag and S. Gueliani, "Event extraction based on deep learning in food hazard arabic texts," 2020, *arXiv:2008.05014*.

[31] A. C. Umarikar and L. Umanand, "Modelling of switching systems in bond graphs using the concept of switched power junctions," *J. Franklin Inst.*, vol. 342, no. 2, pp. 131–147, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0016003204000833

[32] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2017, *arXiv:1609.02907*.

[33] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Li'o, and Y. Bengio, "Graph attention networks," 2018, *arXiv:1710.10903*.

[34] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," 2017. [Online]. Available: https://arxiv.org/abs/1706.02216

[35] M. Schlichtkrull, T. N. Kipf, P. Bloem, R.V.D. Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," 2018, *arXiv:1703.06103*.

[36] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0893608089900208

[37] "Imbg120r045m1h sic mosfet ltspice model," [Online]. Available: https://www.infineon.com/cms/en/product/power/mosfet/silicon-carbide/discretes/imbg120r045m1h/

[38] E. Liberis and N. D. Lane, "Neural networks on microcontrollers: Saving memory at inference via operator reordering," 2020.