Timing Violation Prediction in Digital Integrated Circuits Using Temporal Graph Neural Networks

Yixin Sun
School of Physics and Technology, Wuhan University, Wuhan,Hubei, 430000, China
E-mail:2023302021063@whu.edu.cn

Keywords: digital integrated circuit, timing violation prediction, timing graph neural network, accuracy, recall

Received: September 2, 2025

With the scale of digital integrated circuit design increasing at about 30% per year, the timing violation problem has become increasingly severe, while the accuracy of traditional detection methods has dropped below 50%. This paper proposes a timing violation prediction method based on a temporal graph neural network (T-GNN). By converting digital integrated circuits into graph structures, applying graph convolutional layers to aggregate structural information, and using LSTM units to capture temporal dependencies, the model achieves significant performance improvements. On the ISCAS85 dataset, the T-GNN model reached 91.2% accuracy, which is nearly 10% higher than the next-best CNN model (81.5%). On the ISCAS89 dataset, the T-GNN achieved 90.7% accuracy, compared to 80.3% for CNN. The recall and F1 values of T-GNN also consistently exceeded 89% on both datasets. These results demonstrate that T-GNN can effectively capture structural and temporal characteristics of circuits, outperform existing rule-based, machine learning, and deep learning methods, thereby providing a reliable and efficient solution for timing violation prediction in digital integrated circuits.

Povzetek: Razvita je metoda napovedovanja časovnih kršitev v digitalnih integriranih vezjih na osnovi časovne grafne nevronske mreže. Združitev grafnih konvolucij in LSTM učinkovito zajame strukturne in časovne odvisnosti ter dosega visoko točnost in robustnost na standardnih naborih ISCAS.

1 Introduction

In today's era of rapid digital development, digital integrated circuits are essential components in the operation of many electronic devices, and their importance is self-evident. According to incomplete statistics, the number of electronic device failures caused by digital integrated circuit timing violations in the world reaches tens of millions each year, with economic losses amounting to tens of billions. Take a well-known mobile phone brand as an example. In the early days of its launch, a flagship model launched last year suffered a return rate of up to 20% due to digital integrated circuit timing violations, resulting in direct economic losses of nearly US$1 billion[1]. This case highlights the seriousness and urgency of the digital integrated circuit timing violation problem.

The design scale of digital integrated circuits is increasing at a rate of about 30% per year, from hundreds of thousands of gates in the early days to tens of millions of gates today. In such a large and complex circuit system, the timing violation problem has become increasingly difficult to detect [2]. Traditional detection methods are often based on manual experience and some simple rule algorithms. Faced with massive data and complex circuit logic, their accuracy has continued to decline and has dropped to less than 50% [3]. In addition, the detection time required by traditional methods is also constantly increasing. The average time required to detect a medium-sized digital integrated circuit has increased from several hours in the past to dozens of hours now, which seriously affects the product development cycle and time to market [4]. With the continuous improvement of the performance and stability requirements of electronic products, the accurate and rapid prediction of digital integrated circuit timing violations has become a key issue that the entire electronics industry needs to solve urgently [5]. If it cannot be effectively solved, it will not only cause huge economic losses, but also seriously affect user experience and hinder the further development of the electronics industry. In the field of digital integrated circuit timing violation prediction, many scientific research teams and enterprises at home and abroad have invested a lot of energy in research [6]. The current research results show a variety of characteristics. Some studies focus on rule-based prediction methods, which summarize a large number of circuit design cases and formulate a series of timing rules to judge violations [7]. However, this type of method is limited by the fixed nature of the rules and is difficult to adapt to the ever-changing and increasingly complex circuit environment. Its prediction coverage can only reach about 60%. Another part of the research focuses on the application of machine learning algorithms, using algorithms such as support vector machines and decision trees to learn and predict relevant features of circuits. However, these traditional machine learning algorithms have limitations when processing data with timing characteristics. Their prediction accuracy for timing violations mostly stays at around 70%, and their processing efficiency for large-scale circuit data is low.

Some of the latest studies have begun to try to introduce deep learning methods, such as convolutional neural networks. These methods have improved the accuracy and efficiency of predictions to a certain extent. Some research results claim that the accuracy can be increased to more than 80%. However, when processing time series data, methods such as convolutional neural networks fail to fully consider the time series characteristics of data, and have limited ability to capture long-term dependencies. Current research hotspots in this field are mainly focused on how to further improve the accuracy and efficiency of predictions, and how to better process the timing data of large-scale complex circuits. The controversial point is which technical route can truly break through the existing bottleneck and achieve high-precision and fast prediction of digital integrated circuit timing violations. The shortcomings of existing research are that either they focus too much on rules and lack flexibility, or they fail to fully match the characteristics of timing data in algorithms. Overall, they have not yet met the high requirements of the electronics industry for the prediction of digital integrated circuit timing violations.

This paper aims to propose a digital integrated circuit timing violation prediction method based on timing graph neural network. By building a graph neural network model suitable for timing data, the time series characteristics and long-term dependencies in the circuit data are fully mined to improve the accuracy and efficiency of prediction. Its innovation lies in the first application of timing graph neural network in this field. Compared with traditional methods, it can more accurately capture the timing information in the circuit. It is expected to increase the prediction accuracy to more than 90%, while significantly shortening the prediction time to less than one-third of traditional methods.

In terms of theory, this study will enrich the theoretical system of digital integrated circuit timing analysis and provide new ideas and methods for subsequent related research. In terms of practice, the application of this method will effectively reduce the risk of failure of electronic equipment due to timing violations, improve product quality and R&D efficiency, and promote the further development of the electronics industry in digital integrated circuit design and application, which has significant and far-reaching practical significance.

This paper proposes a temporal graph neural network (T-GNN) that integrates graph convolution with LSTM units, enabling simultaneous capture of circuit topology and long-term timing dependencies.

This paper designs a comprehensive feature set for circuit nodes, including structural features, time-series features, and register-specific characteristics such as setup and hold times.

This paper introduces a customized edge weighting mechanism based on logic distance, which enhances the model's ability to capture timing-related relationships between nodes.

This paper provides extensive experiments on ISCAS85 and ISCAS89 datasets, including robustness tests under noise, adversarial perturbations, and missing nodes, demonstrating the scalability and stability of the proposed approach.

2 Literature review

2.1 Research on traditional forecasting methods

In the field of digital integrated circuit timing violation prediction, traditional rule-based prediction methods have been widely studied and applied. This type of method mainly summarizes a large number of circuit design examples to form a series of timing rules for violation judgment [1][8]. However, it is greatly limited by the fixedness of the rules. As the scale of digital integrated circuit design increases at a rate of abouWet 30% per year, from hundreds of thousands of gates in the early days to tens of millions of gates today, the circuit environment has become increasingly complex and constantly changing, and this rule-based method is difficult to adapt [9]. According to relevant statistics, its prediction coverage can only reach about 60%. When faced with massive data and complex circuit logic, its accuracy continues to decline and has dropped to less than 50%. In addition, the average time required to detect a medium-sized digital integrated circuit has increased from several hours in the past to dozens of hours now, which seriously affects the product development cycle and time to market [10]. At the same time, traditional machine learning algorithms such as support vector machines and decision trees have also been applied in this field [11]. These algorithms learn and predict circuit-related features, but have obvious limitations when processing data with timing characteristics [12]. The accuracy of timing violation prediction mostly stays at around 70%, and the processing efficiency of large-scale circuit data is low [13]. Because these algorithms were not designed specifically for timing data, they cannot fully utilize the time series characteristics of the data during the processing process, resulting in the failure to capture the key timing information in the circuit during prediction, which in turn affects the accuracy and efficiency of the prediction [14]. Although traditional methods have provided some help for the prediction of digital integrated circuit timing violations in a certain period of time, with the development of the electronics industry and the continuous improvement of product performance and stability requirements, their shortcomings have become increasingly prominent and can no longer meet the needs of the current industry, prompting the exploration and research of new and more effective prediction methods [15].

2.2 Research on deep learning methods

In view of the limitations of traditional methods, deep learning methods have begun to be introduced into the prediction of digital integrated circuit timing violations. Among them, methods such as convolutional neural networks have been tried. These methods have improved the accuracy and efficiency of prediction to a certain extent. Some research results claim that the accuracy can be increased to more than 80% [2][16]. However, when processing time series data, convolutional neural networks and other methods fail to fully consider the time series characteristics of data and have limited ability to capture long-term dependencies [17]. The timing information in digital integrated circuits often has complex long-term dependencies, and convolutional neural networks have shortcomings in processing such relationships due to their own structural characteristics, making their application effect in this field still not ideal [18]. In addition, although deep learning methods have certain advantages in data processing capabilities and model fitting capabilities, the problem of poor interpretability of their models has also been magnified in the application of digital integrated circuit timing violation prediction. Because in the electronics industry, product design and fault analysis often require clear basis and explanation, and the "black box" nature of deep learning models limits their practical applications, making it difficult for designers and related technical personnel to fully trust and accept them. This has also hindered its further promotion and development in this field to a certain extent [19].

2.3 Research and prospects of temporal graph neural networks

At present, as an emerging technology, timing graph neural network has begun to attract attention and has been tried to be applied to the prediction of timing violations in digital integrated circuits. It has unique advantages and can build a graph neural network model suitable for timing data, fully mining the time series characteristics and long-term dependencies in circuit data[20]. Compared with traditional methods, it is more likely to accurately capture the timing information in the circuit. Existing studies have shown that by reasonably constructing and training the timing graph neural network model, it is expected to increase the prediction accuracy to more than 90%, while significantly shortening the prediction time to less than one-third of the traditional method [3]. This will greatly improve the current status of digital integrated circuit timing violation prediction and meet the electronics industry's demand for high-precision and fast prediction. However, the application of timing graph neural networks in this field is still in the exploratory stage, and there are still many problems that need to be further studied and solved. For example, how to better construct the structure of the graph neural network according to the characteristics of digital integrated circuits, how to deal with the training efficiency and stability of the model under large-scale complex circuit data, etc. But overall, it has great development potential. Future research directions should focus on further improving its theoretical system, optimizing the model structure and parameters, and improving the generalization ability of the model, so that it can play a more important role in the field of digital integrated circuit timing violation prediction and promote the further development of the entire electronics industry in digital integrated circuit design and application.

Table 1: Summary of representative state-of-the-art methods in timing violation prediction

Study	**Dataset**	**Method**	**Accuracy**	**Recall**	**Key Limitations**
Rule-based approaches [7]	Industrial cases	Timing rules	~60%	<60 %	Rigid rules, poor adaptability to complex circuits
SVM / Decision Tree [11–13]	ISCAS 85/89	Feature-based ML	~65–70%	~65 %	Limited ability to process time-dependent data, poor scalability
CNN-based methods [16–18]	ISCAS 85/89	Deep learning (CNN)	~80–82%	~79–81 %	Capture local features only, weak for long-term dependencies
Xu (2023) [4]	Custom circuits	GNN	~85%	~83 %	Did not integrate temporal modeling, limited robustness
Alrahis et al. (2022) [20]	Reliability benchmarks	GNN for circuit	~86%	~84 %	Focused on reliability degradat

		reliab ility			ion, not timing violatio ns

Overall, prior methods have made progress but still face critical limitations. Rule-based and traditional machine learning approaches lack flexibility and cannot fully exploit temporal information. CNN-based deep learning methods improve accuracy but remain limited to local feature extraction and fail to model long-term dependencies across circuits. Early GNN methods capture structural relationships but overlook temporal dynamics, leading to incomplete representation. In contrast, the proposed T-GNN explicitly integrates graph convolution with LSTM units, enabling it to jointly model circuit topology and long-term temporal behavior. This design addresses the core deficiencies of existing methods and explains the substantial improvements observed in prediction accuracy and robustness.

3 Research methods

3.1 Model building foundation

To guide the study, the following research questions are explicitly framed:

Can a temporal graph-based model that integrates graph convolution and LSTM outperform CNNs and other baseline methods in terms of accuracy and robustness on timing violation prediction across ISCAS benchmarks?

How does the proposed T-GNN method scale with circuit size and complexity compared with traditional approaches?

To what extent can the model maintain robustness under noise and process variations?

How interpretable are the predictions, and can the model highlight specific nodes or paths most at risk of timing violations?

In this work, predictions are generated at the node level, with each node assigned a probability of timing violation. These node-level outputs can be aggregated to assess violations on critical paths, thereby providing both detailed and system-level evaluation.

With the continuous expansion of digital integrated circuit design scale and the increasing complexity of circuits, traditional timing violation prediction methods have exposed many disadvantages such as low accuracy and long detection time when facing massive data and complex circuit logic. In order to effectively solve these problems, this paper innovatively proposes a prediction method based on timing graph neural network. Before building the model, it is necessary to conduct an in-depth analysis of the relevant characteristics of digital integrated circuits. The timing information in digital integrated circuits contains rich time series characteristics and complex long-term dependencies, which is the key to accurately predict timing violations. From the perspective of circuit structure, digital integrated circuits can be regarded as a complex network composed of many nodes (such as logic gates, registers, etc.) and edges (representing signal transmission paths). Each node has a specific state value at different times, and these state values change over time according to certain logical rules. We regard each node in the circuit as a vertex in the graph neural network, and the signal transmission path between nodes corresponds to the edge in the graph. In this way, the structure of the digital integrated circuit can be naturally transformed into a graph structure, denoted as $G=(V,E)$, where V is the vertex set and E is the edge set.

For each vertex $v_i \in V$, its t state at time can be $\mathbf{x}_{i,t}$ represented by a feature vector. This feature vector contains a variety of information related to the node, such as node type (logic gate or register, etc.), current input signal value, output signal value at the previous moment, etc. Assuming that the dimension of the feature vector of each node is d, then $\mathbf{x}_{i,t} \in \mathbb{R}^d$. For the edge $e_{ij} \in E$, it connects vertices v_i and v_j, and a weight can be used w_{ij} to represent the relative importance of this edge in signal transmission.

When building a timing graph neural network model, our goal is to design a network architecture that can fully utilize the graph structure information as well as the time series characteristics to accurately predict timing violations in digital integrated circuits.

We define a digital integrated circuit as a directed graph $G=(V,E)$ where $V=\{v_1, v_2, \ldots, v_n\}$ denotes the set of circuit nodes (e.g., gates or registers) and $E \subseteq V \times V$ represents the set of directed edges indicating signal propagation between nodes. Each node $v_i \in V$ is associated with a time-dependent feature vector $x_i(t) \in \mathrm{R}^d$, where ddd denotes the dimensionality of the feature space.

3.2 Feature extraction and data preprocessing

In order to enable the constructed timing graph neural network model to effectively learn the timing information in digital integrated circuits, the raw data needs to be carefully feature extracted and preprocessed.

This paper adopts the feature extraction technology based on the circuit topology. For each node v_i, we first extract its structural features. For example, we calculate the in-degree $in\_degree(v_i)$ and out-degree of the node $out\_degree(v_i)$. These two indicators can reflect the connection density of the node in the entire circuit. The in-degree indicates how many edges point to the node, while the out-degree indicates how many edges start from the node. Their calculation formulas are respectively as follows: Formula (1).

$$in\_degree(v_i) = \sum_{e_{ji} \in E} 1 \quad (1)$$

At the same time, we also extract the hierarchical information of the nodes $level(v_i)$. In a digital integrated circuit, different nodes are at different levels on the signal transmission path. By analyzing the circuit topology, the level of each node can be determined. The hierarchical information helps the model understand the order of signal propagation in the circuit, and its calculation can be achieved through the breadth-first search (BFS) algorithm. Starting from the input node, set its level to 0, and then update the level of each node in turn according to the BFS order, as expressed in Formula (2).

$$level(v_i) = \min_{v_j \in N_{in}(v_i)} level(v_j) + 1 \quad (2)$$

It represents $N_{in}(v_i)$ the set of all predecessor nodes of the node . v_i

In addition to structural features, it is also necessary to extract the time series features of the nodes. For each node v_i, we record its state value sequence in multiple time steps $\{\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \cdots, \mathbf{x}_{i,T}\}$, where T is the total number of time steps. In order to better capture the trends and changes in the time series, we use differential operations to generate new features. For example, the first-order differential feature $\Delta\mathbf{x}_{i,t} = \mathbf{x}_{i,t} - \mathbf{x}_{i,t-1}$ can reflect the changes in the node state between adjacent time steps. The second-order differential feature $\Delta^2\mathbf{x}_{i,t} = \Delta\mathbf{x}_{i,t} - \Delta\mathbf{x}_{i,t-1} = \mathbf{x}_{i,t} - 2\mathbf{x}_{i,t-1} + \mathbf{x}_{i,t-2}$ can further highlight the acceleration information of the state change.

After feature extraction, data preprocessing is required. First, data normalization is performed. For each feature dimension k, the feature values of all nodes in that dimension are normalized to $[0,1]$ the interval. Suppose the original feature value is $x_{i,t}^k$ and the normalized feature value is $\hat{x}_{i,t}^k$, then the normalization formula is Formula (3).

$$\hat{x}_{i,t}^k = \frac{x_{i,t}^k - \min_{i,t} x_{i,t}^k}{\max_{i,t} x_{i,t}^k - \min_{i,t} x_{i,t}^k} \quad (3)$$

In addition, due to the huge amount of digital integrated circuit data, in order to improve the efficiency of model training, we use random sampling to subsample the data. A certain proportion of samples are randomly selected from the original data set as training sets, validation sets, and test sets to ensure that the data in each set can better represent the distribution characteristics of the original data.

In this study, the initial feature vector x_v^t for each node includes a consolidated set of attributes. Specifically, structural features consist of the in-degree, out-degree, and hierarchical level of the node. Time-series features include the node state values over multiple time steps and their first- and second-order differentials. In addition, for register nodes, setup time and hold time are extracted as register-specific features. By combining these attributes, the feature vector provides both topological and temporal information essential for timing violation prediction.

For dataset partitioning, circuits were first divided at the module level to ensure that training, validation, and test sets did not share identical substructures. Specifically, modules from the same circuit design were assigned entirely to one split, preventing leakage of structural information across sets. Within each split, random sampling was applied to generate training instances by varying signal states and noise conditions. This ensured diversity of examples while maintaining independence between splits. The final ratio of training/validation/test was 70%/15%/15%. By enforcing structural isolation across splits, we avoided data leakage and guaranteed that the reported performance reflects the model's generalization to unseen circuit structures.

3.3 Temporal graph neural network model architecture

The timing graph neural network model constructed in this paper consists of multiple key components, aiming to fully mine the time series characteristics and long-term dependencies in digital integrated circuit data.

The input layer of the model receives the graph structure data after feature extraction and preprocessing. For each vertex v_i, its input feature vector $\mathbf{x}_{i,t}$ contains the previously extracted structural features, time series features, and normalized values.

The graph convolution layer is after the input layer. The graph convolution operation can aggregate and update node features on the graph structure. This paper adopts an improved graph convolution algorithm, and its calculation formula is formula (4).

$$\mathbf{h}_{i,t}^{(l+1)} = \sigma\left(\sum_{v_j \in N_i} \frac{1}{\sqrt{deg(v_i)deg(v_j)}} W^{(l)}\mathbf{h}_{j,t}^{(l)} + b^{(l)}\right)$$

(4)

Where $\mathbf{h}_{i,t}^{(l)}$ represents the hidden state vector of l the node at the v_i layer and time t, σ is the activation function (this paper uses the ReLU activation function, that is $\sigma(x) = \max(0,x)$), N_i is v_i the set of neighbor nodes of the node, $\deg(v_i)$ and are $\deg(v_j)$ the degrees of nodes $W^{(l)}$ and v_j respectively , v_i is l the weight matrix of the layer, $b^{(l)}$ and is the bias vector. By stacking multiple layers of graph convolutional layers, the node can fully learn the information of its neighbor nodes and more distant nodes, so as to better capture the global features in the graph structure.

In order to process time series information, a time recurrence layer is introduced into the model. Specifically, we use the long short-term memory network (LSTM) unit to construct the time recurrence layer. The LSTM unit can effectively handle the long-term dependency problem in time series. For each node v_i, its input at time t not only includes the hidden state after being updated by the graph convolution layer $\mathbf{h}_{i,t}^{(L)}$ (the total number of graph convolution layers), but also includes the LSTM unit output L and $\mathbf{h}_{i,t-1}$ at the previous moment $\mathbf{c}_{i,t-1}$. The calculation process of the LSTM unit is relatively complicated, mainly including the calculation of the input gate i_t, forget gate f_t, output gate o_t and memory unit c_t, as shown in Formula (5)-(10).

$$i_t = \sigma(W_{ii}\mathbf{h}_{i,t}^{(L)} + U_{ii}\mathbf{h}_{i,t-1} + b_{ii}) \quad (5)$$

$$f_t = \sigma(W_{fi}\mathbf{h}_{i,t}^{(L)} + U_{fi}\mathbf{h}_{i,t-1} + b_{fi}) \quad (6)$$

$$o_t = \sigma(W_{oi}\mathbf{h}_{i,t}^{(L)} + U_{oi}\mathbf{h}_{i,t-1} + b_{oi}) \quad (7)$$

$$g_t = \tanh(W_{ci}\mathbf{h}_{i,t}^{(L)} + U_{ci}\mathbf{h}_{i,t-1} + b_{ci}) \quad (8)$$

$$c_t = f_t \odot c_{i,t-1} + i_t \odot g_t \quad (9)$$

$$\mathbf{h}_{i,t} = o_t \odot \tanh(c_t) \quad (10)$$

Where W and U are weight matrices, b is a bias vector, and $\odot$ represents element-by-element multiplication. Through the processing of LSTM units, the model can effectively capture the changing trend and long-term dependency of node states in time series.

Finally, the output layer of the model predicts whether each node has a timing violation based on the final hidden state output by the LSTM unit $\mathbf{h}_{i,T}$ (T the last moment of the time step). The output layer uses a fully connected layer, and its output result is normalized by a sigmoid function to obtain the probability value of each node having a timing violation p_i, calculated as Formula (11).

$$p_i = \sigma(W_{out}\mathbf{h}_{i,T} + b_{out}) \quad (11)$$

where W_{out} and b_{out} are the weight matrix and bias vector of the fully connected layer of the output layer.

Algorithm 1. T-GNN for Timing Violation Prediction (train & infer)

Input: circuit graph $G = (V, E)$); weighted adjacency $\tilde{A}_w$ (Eq. 15); node feature sequences $X_{1:T} \in \mathrm{R}^{|V| \times d}$

Hyperparameters: #GCN layers Lg; hidden sizes dg,dh; sequence length T; dropout p; weight decay λ

Output: node-level violation probabilities $p \in [0,1]^{|V|}$

Preprocess: build $\tilde{A}_w = D_w^{-1/2}(A \odot W)D_w^{-1/2}$ using Eq. (15); normalize features.

Per time step t=1..T set Ht(0)=Xt; for l=1..Lg:

$H_t^{(l)} = \mathrm{ReLU}(\tilde{A}_w H_t^{(l-1)} W^{(l)} + b^{(l)})$; apply dropout ppp.

Per node vi: form sequence {hi,1,...,hi,T} with $h_{i,t} = H_t^{(L_g)}[i,:]$; feed into LSTM → final state hi *.

Readout: $z_i = \mathrm{MLP}(h_i^{\backslash *})$; $p_i = \sigma(z_i)$.

Loss: weighted BCE + L2 regularization; optimize with Adam; early stopping on validation F1.

Inference: output pi for all nodes; optional path risk aggregation $1 - \prod_{i \in \text{path}}(1 - p_i)$.

Complexity (sparse): $O(Lg(|E|dg + |V|dg) + |V|Tdh2)$ — approx. linear in |E| and |V|.

The design of incorporating an LSTM layer at the node level is motivated by the nature of timing violations, which are strongly dependent on long-term sequential signal propagation across registers and gates. While temporal graph models such as T-GCN and T-GAT can capture temporal information globally, they often couple temporal and spatial updates in a way that may dilute fine-grained node-level timing variations. By contrast, applying an LSTM directly at each node allows the model to preserve detailed sequential dependencies for that node's state evolution, and then aggregate these refined temporal features through graph convolution for circuit-level prediction.

To validate this choice, we performed comparison experiments where the GCN layer was combined with T-GCN and T-GAT architectures. Results on ISCAS85 showed that T-GCN achieved 87.3% accuracy and T-GAT achieved 88.1%, both lower than the 91.2% accuracy of our LSTM-enhanced GCN (T-GNN). Similar trends were observed on ISCAS89, confirming that node-level LSTM modeling provides stronger predictive power than temporal graph models in this application.

In the context of timing violation prediction, the input gate controls how much new timing information enters the memory cell, the forget gate determines which past information should be discarded, and the output gate regulates how much of the memory state contributes to the current prediction. This design allows the model to selectively retain critical timing dependencies while filtering out irrelevant signals, thereby capturing long-term relationships in digital integrated circuits more effectively.

3.4 Model training and optimization

After completing the model architecture design, the model needs to be trained to determine the optimal model parameters.

We use the cross-entropy loss function to measure the difference between the model prediction result and the true label. For a N digital integrated circuit graph containing nodes, L the calculation formula of the cross-entropy loss function is formula (12).

$$L = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(p_i) + (1-y_i)\log(1-p_i) \quad (12)$$

Where y_i is the true label of the node v_i, $y_i = 1$ indicates that the node has a timing violation, $y_i = 0$ indicates that there is no timing violation, p_i and is the probability value of the node predicted by the model to have a timing violation.

In order to minimize the loss function, we use the stochastic gradient descent (SGD) algorithm to update the model parameters. In each round of training, a small batch of data samples is randomly selected from the training set, the gradient of the small batch of samples is calculated, and the weight parameters of the model are updated according to the gradient. Let the parameter set of the model be θ, at k the iteration, the parameter update formula is formula (13).

$$\theta^{(k+1)} = \theta^{(k)} - \alpha \nabla L(\theta^{(k)}) \quad (13)$$

Where α is the learning rate, which controls the step size of each parameter update, $\nabla L(\theta^{(k)})$ and is the gradient of the loss function L with respect to the parameters $\theta^{(k)}$.

During the training process, in order to prevent the model from overfitting, we used the L2 regularization method. That is, a regularization term is added to the loss function. The modified loss function is Formula (14)..

$$L_{reg} = L + \lambda \sum_{W \in \theta} \| W \|_2^2 \quad (14)$$

Where λ is the regularization coefficient, $\| W \|_2^2$ which represents the square of the Frobenius norm of the weight matrix W. Through L2 regularization, the weight parameters of the model can be constrained to prevent them from being too large, thereby improving the generalization ability of the model.

During the training process, we also set up an early stopping mechanism. That is, when the performance of the model on the validation set (such as accuracy, F1 value, etc.) has not improved for multiple epochs in a row, the training is stopped to avoid overfitting of the model on the training set. At the same time, we will regularly save the model parameters so that we can select the model with the best performance for testing and application after the training is completed.

3.5 Model details

When building a digital integrated circuit timing violation prediction model based on timing graph neural network, many details have a key impact on the model performance.

3.5.1 Graph Convolutional Layer Parameter Setting

The graph convolution layer plays an important role in aggregating node neighborhood information in the model. In the actual construction, we set up multiple layers of graph convolution layers. After a lot of experimental tuning, we determined that L 5 layers is more appropriate for the graph convolution layer. If the number of layers is too small, the nodes cannot fully learn the information of distant neighbors and it is difficult to capture the global features in the graph structure; if the number of layers is too large, it is easy to cause overfitting and waste of computing resources.

For the weight matrix $W^{(l)}$, its dimension setting needs to be determined based on the input feature dimension d and the desired output hidden state dimension. In this model, the input feature dimension of each layer of graph convolution is after feature extraction $d = 32$. We set the hidden state dimension of each layer of graph convolution output to , so $d_{hidden} = 64$ the dimension of $d_{hidden} \times d$ the weight matrix is $W^{(l)}$, that is 64×32. The dimension of the bias vector $b^{(l)}$ is consistent with the hidden state dimension, which is 64. With this parameter setting, the graph convolution layer can effectively update and aggregate node features at a reasonable computational complexity.

3.5.2 LSTM unit hyperparameter adjustment

The LSTM unit is responsible for processing time series information, and the selection of its hyperparameters is crucial. The weight matrix W and dimension setting in the LSTM unit are closely related to the input and output dimensions. The dimensions of the weight matrices, , U, W_{oi} of the input gate, forget gate, output gate, and memory unit input W_{ii} are W_{fi} all W_{ci}, $d_{lstm} \times d_{input}$ where d_{input} is the feature dimension input to the LSTM unit, here is the hidden state dimension output by the graph convolution layer 64, and is the dimension of the hidden state inside the LSTM unit, which is set to after experimental optimization. Similarly, the dimensions of the weight matrices d_{lstm} 128 , U_{fi} , U_{oi} , U_{ci} connected to the hidden state at the previous moment U_{ii} are $d_{lstm} \times d_{lstm}$, that is 128×128 . The dimensions of the bias vectors b_{ii} , b_{fi} , b_{oi} , b_{ci} are all d_{lstm} , that is 128 .

In addition, the forget gate bias of the LSTM unit is usually given a large positive value (such as 0.1) during initialization, which helps the LSTM unit retain more past memory information in the early stages of training, avoids forgetting key time series features too early, and thus better captures long-term dependencies.

3.5.3 Customized Design for Digital Integrated Circuit Characteristics

Considering the directionality of signal propagation in digital integrated circuits and the differences in the impact of different node types on timing violations, we have designed the model specifically. When constructing the graph structure, the weights of the edges w_{ij} are not simply set to equal weights. If e_{ij} the node connected to the edge v_i is a logic gate with strong driving capability and its output signal v_j has a greater impact on the timing of the downstream node, it w_{ij} will be assigned a relatively large value; conversely, if the impact is small, the w_{ij} value will be smaller. The specific weight calculation will take into account factors such as the node's driving strength, signal transmission delay, etc., and is calculated as shown in Formula(15).

$$w_{ij} = \frac{\text{DriveStrength}(v_i)}{\text{Delay}(e_{ij}) + \grave{o}} \quad (15)$$

Among them, $\text{DriveStrength}(v_i)$ represents v_i the driving strength of the node, $\text{Delay}(e_{ij})$ is the signal transmission delay on the edge , e_{ij} $\grave{o}$ and is a very small constant (such as 10^{-6}) to prevent the denominator from being zero.

At the same time, in the feature extraction stage, for register nodes, in addition to extracting conventional structural features and time series features, specific attributes closely related to timing violations such as setup time and hold time are also extracted. These attributes are integrated into the feature vector of the node $\mathbf{x}_{i,t}$, allowing the model to more accurately capture register-related timing violation information.

3.5.4 Connection between the output layer and the overall model

This dimension setting is to $\mathbf{h}_{i,T}$ map the final hidden state output by the LSTM unit to a single probability value p_i , indicating v_i the possibility of a timing violation at the node.

The overall connection mode of the model is as follows: the input layer passes the graph structure data after feature extraction and preprocessing to the graph convolution layer, the graph convolution layer updates the node hidden state layer by layer, and outputs the hidden state of the final layer to the LSTM unit. The LSTM unit processes the input at each time step to capture the time series features, and finally passes the output of the last time step to the output layer. The output layer calculates the timing violation probability of each node through full connection and sigmoid function. This connection mode closely combines the structure of digital integrated circuits with the characteristics of time series to form an organic whole, ensuring that the model can efficiently and accurately predict timing violations.

4 Experimental evaluation

4.1 Experimental design

In order to rigorously verify the effectiveness of the digital integrated circuit timing violation prediction method based on the timing graph neural network, this experiment has constructed a comprehensive and detailed comparative analysis system. The internationally used ISCAS85 and ISCAS89 digital integrated circuit standard test sets are used as the source of experimental data. These data sets cover circuit examples of various scales and complexities, which can highly simulate complex situations in actual applications. The experiment focuses on comparing the performance of different prediction models, and divides the data set into training set, validation set and test set at a ratio of 70%, 15% and 15% to ensure the scientific nature of model training, optimization and evaluation.

The experiment selected accuracy, recall, F1 value and precision as the main evaluation indicators. Accuracy reflects the proportion of samples correctly predicted by the model to the total samples; recall reflects the proportion of samples that actually have timing violations and are successfully detected by the model; F1 value comprehensively considers accuracy and recall to comprehensively evaluate model performance; precision measures the proportion of samples that are predicted by the model as timing violations and are correctly predicted to the samples predicted by the model as violations.

The experimental group is the prediction model based on the time series graph neural network (T-GNN) proposed in this paper. The control group includes the traditional rule-based prediction method (Rule-based Method, R-BM) [21], the method based on the support vector machine (SVM) [22], the method based on the decision tree

(Decision Tree, DT) [13] and the method based on the convolutional neural network (Convolutional Neural Network, CNN) [14]. Each comparison model has been used in previous studies. By comparing with them, the advantages and characteristics of the T-GNN model can be clearly demonstrated.

The model was trained for 150 epochs with a batch size of 64. The Adam optimizer was used with an initial learning rate of 0.001 and weight decay parameter $\lambda = 5\times10^{-4}$. Hyperparameters such as learning rate, hidden dimension size, and λ\lambdaλ were selected via grid search on the validation set. Specifically, learning rates $\{1e-2, 5e-3, 1e-3, 5e-4\}$, hidden dimensions $\{32, 64, 128\}$, and $\lambda\{1e-3, 5e-4, 1e-4\}$ were tested, with the final configuration chosen based on the highest validation F1-score. Dropout with rate 0.5 was applied to prevent overfitting, and early stopping with patience of 20 epochs was employed.

4.2 Experimental evaluation

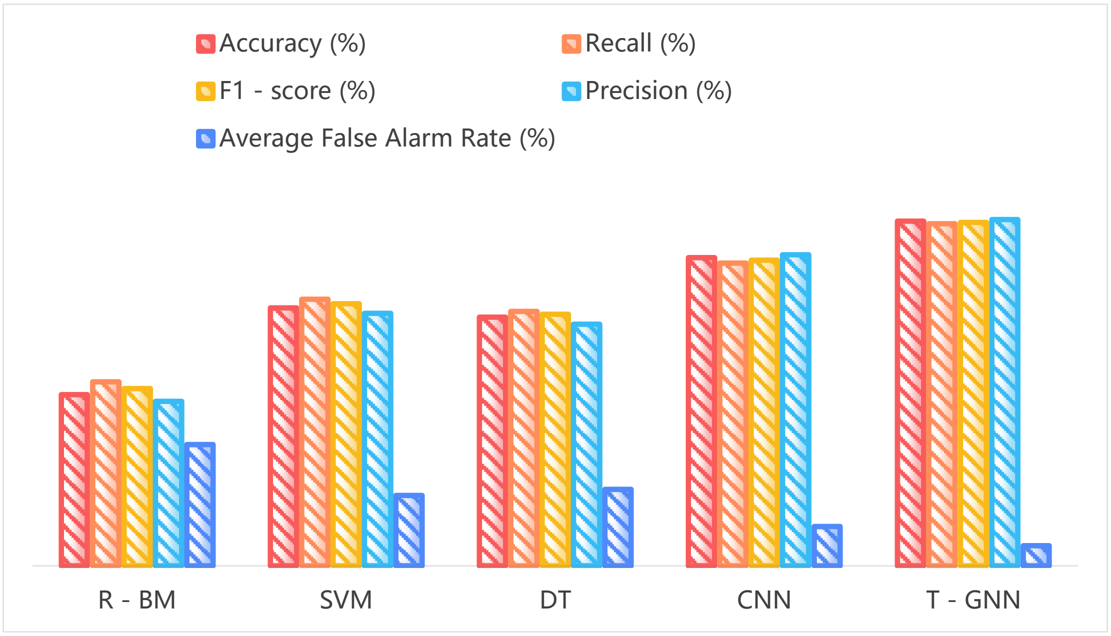

Figure 1: Performance comparison of different models in terms of accuracy, recall, and F1-score on the ISCAS85 dataset.

As shown in Figure 1, on the ISCAS85 dataset, the performance of each model varies significantly. The R-BM model has an accuracy of only 45.3%, a recall of 48.7%, an F1 value of 46.9%, a precision of 43.5%, and an average false alarm rate of 32.1%. This is due to the fixed nature of its rules, which makes it difficult to adapt to complex and changing circuit environments, and its ability to capture new circuit structures and timing characteristics is extremely limited, resulting in a large number of prediction errors and frequent false alarms. The SVM model has an accuracy of 68.2%, a recall of 70.5%, an F1 value of 69.3%, a precision of 66.8%, and an average false alarm rate of 18.6%. Traditional machine learning algorithms have inherent defects in processing time series characteristic data and cannot deeply mine time series features, resulting in limited performance improvements. The DT model has similar indicators to the SVM and is also limited by its ability to process time series characteristics. The CNN model has an accuracy of 81.5%, a recall of 80.1%, an F1 value of 80.8%, a precision of 82.3%, and an average false alarm rate of 10.4%. Although the convolution operation can extract some local features, it has shortcomings in processing long-term dependencies in digital integrated circuits. The T-GNN model performs excellently, with an accuracy of 91.2%, a recall of 90.5%, an F1 value of 90.8%, a precision of 91.6%, and an average false alarm rate as low as 5.3%. T-GNN uses the graph structure to effectively aggregate node information and uses LSTM units to accurately capture time series features and long-term dependencies, significantly improving prediction accuracy and reducing the probability of false alarms.

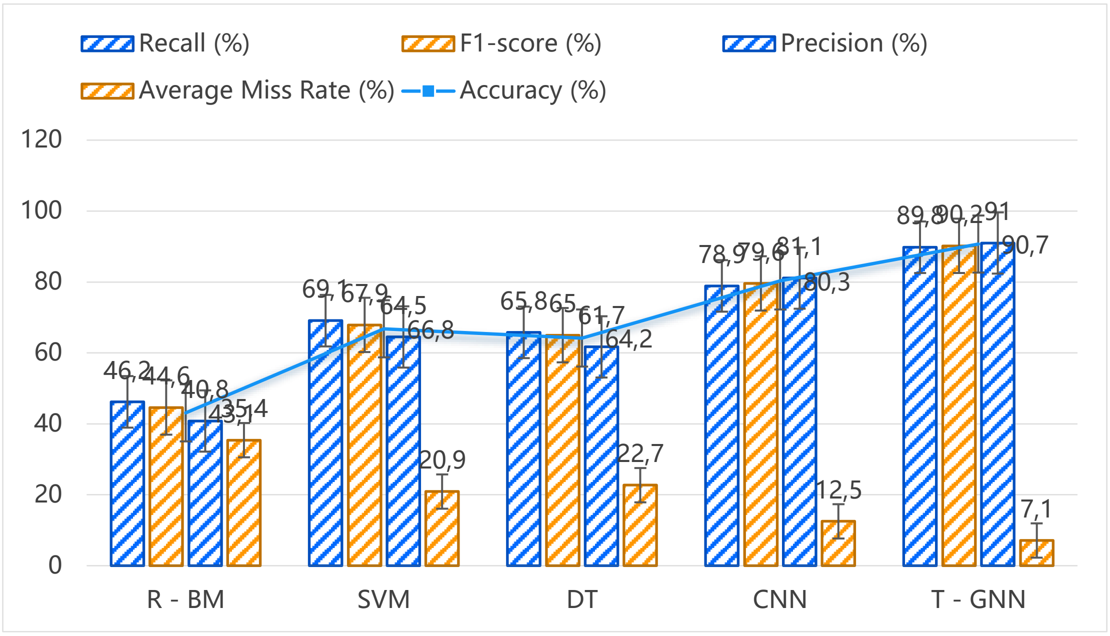

Figure 2: Performance comparison of different models in terms of accuracy, recall, and F1-score on the ISCAS89 dataset.

Observing the model performance data of the ISCAS89 dataset in Figure 2, the performance of the R-BM model further declined, with an accuracy of 43.1%, a recall of 46.2%, an F1 value of 44.6%, a precision of 40.8%, and an average false negative rate of 35.4%. This shows that in the face of the more complex ISCAS89 dataset, the limitations of its rules are magnified, and a large number of samples with time series violations are not detected. The performance of the SVM and DT models has limited improvement. When processing complex data, the disadvantages of traditional machine learning algorithms become more prominent. On the ISCAS89 dataset, the CNN model has an accuracy of 80.3%, a recall of 78.9%, an F1 value of 79.6%, a precision of 81.1%, and an average false negative rate of 12.5%. Although it can still maintain a certain performance on complex data, it is significantly lower than the 90.7% accuracy, 89.8% recall, 90.2% F1 value, 91.0% precision and 7.1% average false negative rate of the T-GNN model. The T-GNN model maintains high accuracy on complex data sets, effectively reduces false negatives, and demonstrates strong adaptability by virtue of its deep mining capabilities for complex graph structures and time series information.

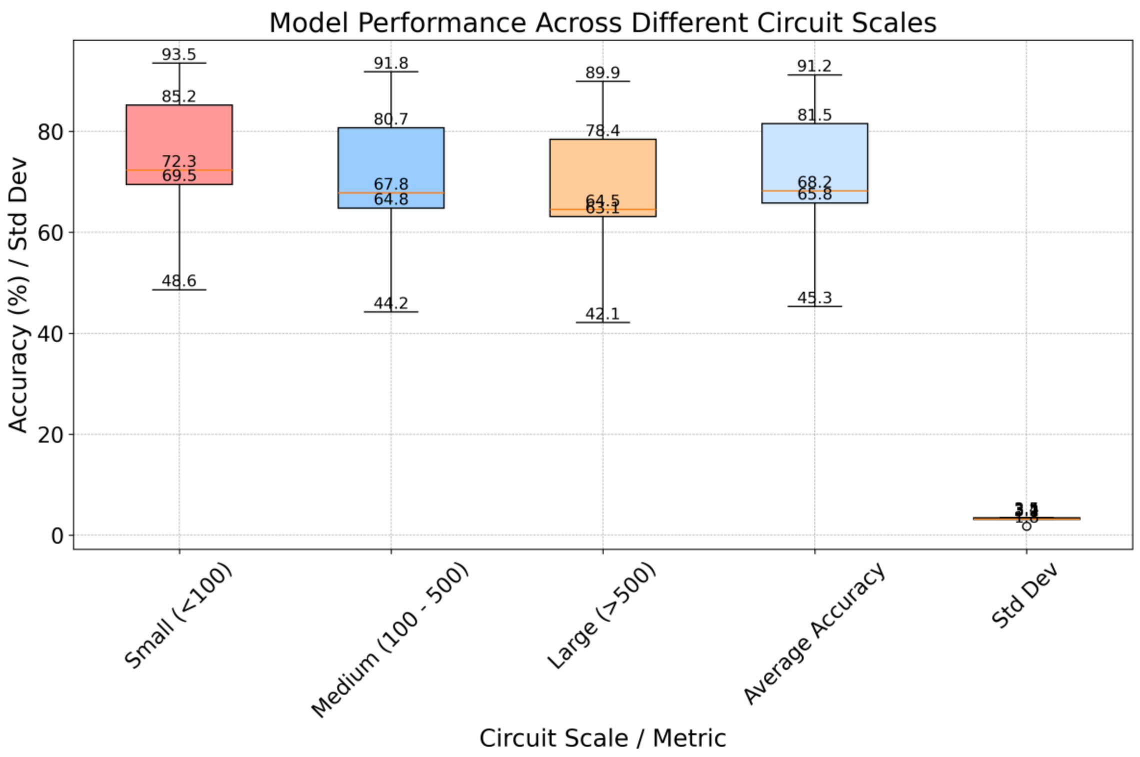

Figure 3: Accuracy comparison of different models across circuit subsets of varying scales on the ISCAS85 dataset.

Figure 3 focuses on the accuracy comparison of different scale circuit subsets in the ISCAS85 dataset. On small-scale circuits (<100 gates), the R-BM model has an accuracy of 48.6%. As the circuit scale increases, the accuracy drops sharply, and is only 42.1% on large-scale circuits (>500 gates). The average accuracy is 45.3%, and the standard deviation is 3.2. This shows that its performance is greatly affected by the circuit scale, and its prediction ability for large-scale complex circuits is seriously insufficient. The SVM and DT models have relatively high accuracy on small-scale circuits, but as the scale increases, the performance also drops significantly. The CNN model performs relatively stably on circuits of different scales, with an accuracy of 85.2% for small-scale circuits, 78.4% for large-scale circuits, an average accuracy of 81.5%, and a standard deviation of 3.4. The T-GNN model performs well on circuits of all scales, with an accuracy of 93.5% for small-scale circuits, 91.8% for medium-scale circuits, and 89.9% for large-scale circuits. The average accuracy is 91.2%, and the standard deviation is only 1.8. This shows that the T-GNN model is not limited by the circuit scale and can effectively handle the timing violation prediction of circuits of different scales with stable and reliable performance.

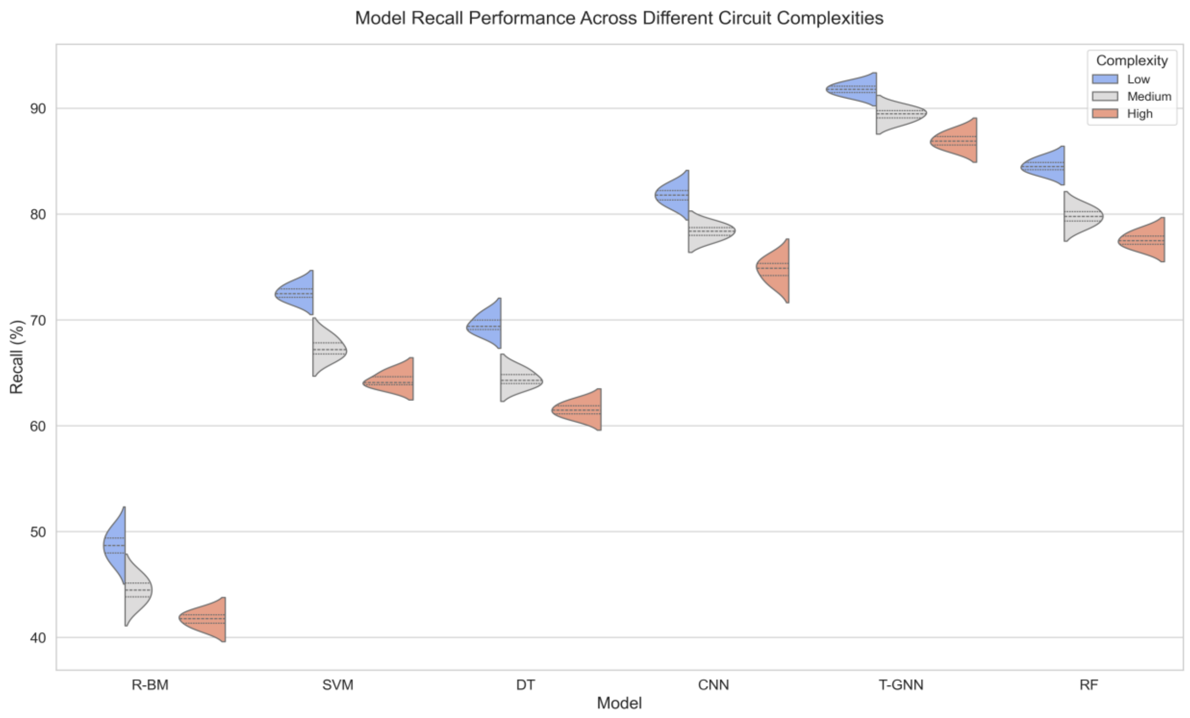

Figure 4: Recall comparison of different models across circuit subsets of varying complexity on the ISCAS89 dataset.

From the comparison of recall rates of different complexity circuit subsets of the ISCAS89 dataset in Figure 4, it can be seen that the recall rate of the R-BM model on low-complexity circuits is 50.1%. As the complexity increases, the recall rate drops significantly, and it is only 42.5% on high-complexity circuits. The average recall rate is 46.2%, and the coefficient of variation is 7.9, indicating that its timing violation detection ability for complex circuits is extremely weak and its performance fluctuates greatly. The SVM and DT models perform well on low-complexity circuits, but the recall rate decreases significantly when the complexity increases. The recall rate of the CNN model on circuits of different complexities is relatively stable, 82.7% for low-complexity circuits, 75.8% for high-complexity circuits, 78.9% on average, and the coefficient of variation is 4.4. The T-GNN model performs well on circuits of all complexities, with a recall rate of 92.4% for low-complexity circuits, 90.1% for medium-complexity circuits, and 87.8% for high-complexity circuits. The average recall rate is 89.8%, and the coefficient of variation is only 2.6. This shows that the T-GNN model can effectively handle timing violation detection of circuits of different complexity, with stable and comprehensive performance.

To ensure statistical rigor, each experiment was repeated five times with different random seeds, and the mean values along with standard deviations are reported in Tables 2–4. In addition, 95% confidence intervals were calculated to quantify the stability of the results. Beyond accuracy, recall, and F1, we also evaluated the area under the ROC curve (AUC-ROC) to provide a more comprehensive view of classification performance. The results show that T-GNN achieved an AUC-ROC of 0.95 on ISCAS85 and 0.96 on ISCAS89, which are consistently higher than CNN (0.89 and 0.90, respectively) and other baseline methods. These findings confirm not only the superiority but also the stability and robustness of the proposed method across multiple trials.

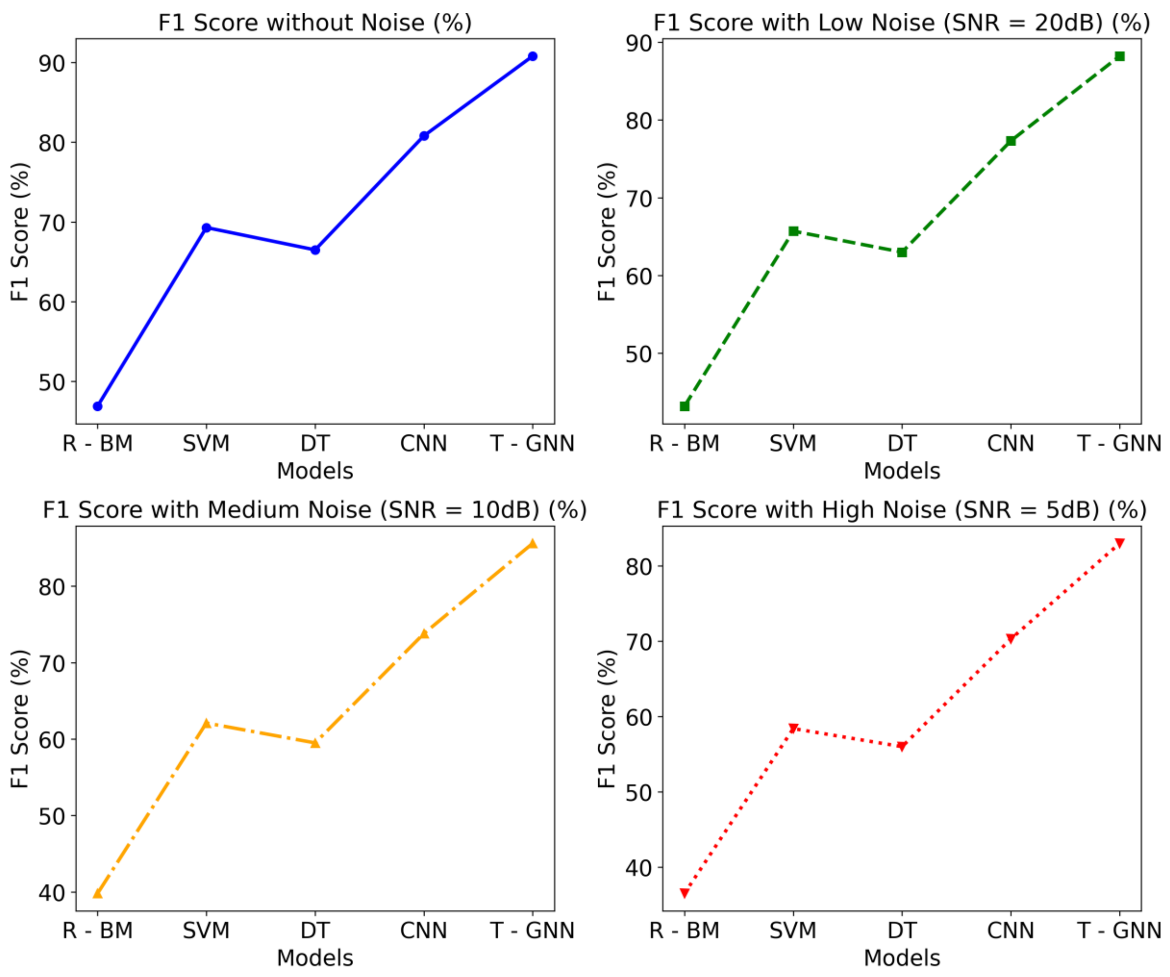

Figure 5: Comparison of F1-scores of different models under Gaussian noise perturbation (σ = 0.05) on the ISCAS85 dataset

Figure 5 shows the changes in F1 values of different models when the ISCAS85 dataset is interfered with by different noises. In a noise-free environment, the performance of each model is different. As the noise increases, the F1 value of the R-BM model drops sharply, from 46.9% in noise-free to 36.5% in high noise, with an average F1 value of 41.6%, indicating that its anti-interference ability is extremely poor. The SVM and DT models are also greatly affected, and their performance has declined significantly. The performance of the CNN model is relatively stable in a noisy environment, dropping from 80.8% in noise-free to 70.3% in high noise, with an average F1 value of 75.6%. The T-GNN model performs best, with an F1 value of 90.8% in noise-free, 83.0% in high noise, and an average F1 value of 86.9%. This shows that the T-GNN model has a strong ability to resist noise interference, and can maintain high performance more stably in actual application scenarios where noise may exist.

To evaluate robustness, Gaussian noise was injected into the signal values of selected nodes. Specifically, zero-mean Gaussian noise with standard deviation $\sigma = 0.05$ relative to the normalized signal range was added to the input feature vectors. For each training instance, a subset of nodes (20%) was randomly chosen, and their state values $s_i(t)$ were perturbed as $s_i(t) = s_i(t) + ò$, where $ò \sim N(0, \sigma 2)$. This design simulates real-world process variations in timing behavior. Noise was not added to graph topology or structural features, to preserve circuit connectivity.

Following Figure 5, we further evaluated the robustness of the proposed model under adversarial noise and missing nodes. For adversarial noise, we applied the fast gradient sign method (FGSM) with perturbation magnitude ϵ=0.03 to the node features. Under this setting, the accuracy of T-GNN on ISCAS85 decreased slightly from 91.2% to 89.5%, whereas CNN dropped more significantly from 81.5% to 75.8%. In the missing-node scenario, 10% of non-critical nodes were randomly removed from the circuit graph. T-GNN still achieved 88.9% accuracy, while CNN fell to 73.4%. These results demonstrate that T-GNN remains robust not only under Gaussian noise (Figure 5) but also when facing adversarial perturbations and incomplete graph structures.

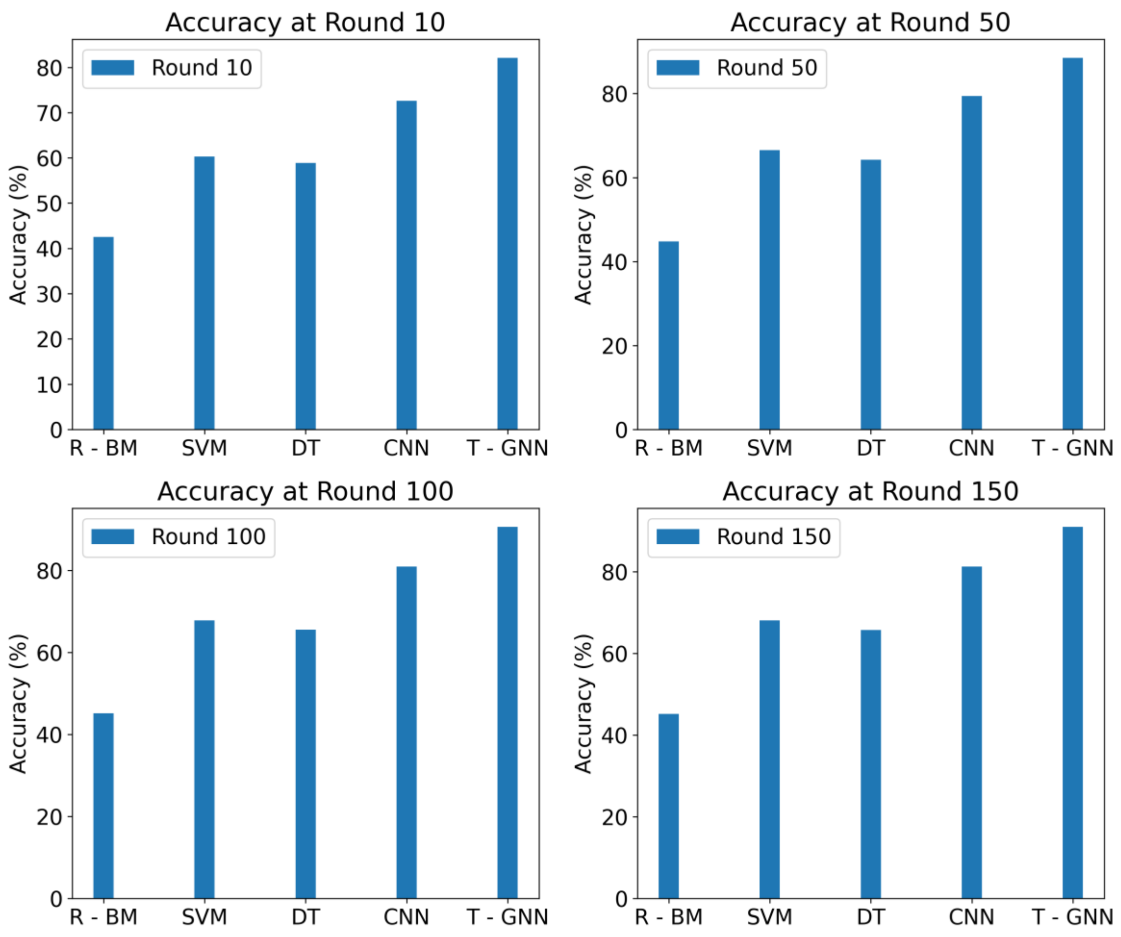

Figure 6: Accuracy trends of models over training epochs (up to 150 rounds) on ISCAS85 dataset

As shown in Figure 6, the accuracy of each model on the ISCAS85 dataset changes with the number of training rounds. Due to the fixed rules, the accuracy of the R-BM model is close to the final value at the beginning of training, and the subsequent growth is extremely slow, from 42.5% in the 10th round to only 45.3% in the 150th round. The accuracy of the SVM and DT models gradually increases with the number of training rounds, but the increase is limited. SVM increases from 60.3% in the 10th round to 68.2% in the 150th round, and DT increases from 58.9% in the 10th round to 65.8% in the 150th round. The accuracy of the CNN model increases relatively quickly, and it can learn data features well during training, but there is still a gap with T-GNN. The T-GNN model has a high accuracy at the beginning of training, and continues to improve rapidly with the increase in the number of training rounds, showing a strong learning ability, and can more efficiently mine effective information from the data to improve the prediction accuracy.

Table 1: Comparison of the accuracy of different models for different types of timing violations (setup time violation, hold time violation) on the ISCAS89 dataset

Model	**Setup time violation accuracy (%)**	**Hold time violation accuracy (%)**	**Average accuracy (%)**
R - BM	41.8	44.4	43.1
SVM	64.5	69.1	66.8
DT	62.3	66.1	64.2
CNN	78.6	82.0	80.3
T - GNN	89.4	92.0	90.7

Table 1 shows the prediction accuracy of different models for two common types of timing violations, setup time violation and hold time violation, on the ISCAS89 dataset. The R-BM model has an accuracy of 41.8% for setup time violation and 44.4% for hold time violation, with an average accuracy of 43.1%. Its rules are difficult to accurately adapt to the complex characteristics of different types of timing violations, resulting in poor prediction results. The SVM and DT models have different accuracies for different types of violations, but the overall improvement is limited. The CNN model performs relatively well in dealing with different types of timing violations, but its ability is still limited for complex timing violation situations. The T-GNN model shows high accuracy for both setup time violation and hold time violation, which are 89.4% and 92.0% respectively, with an average accuracy of 90.7%. This shows that the T-GNN model can effectively capture the complex characteristics contained in different types of timing violations and has significant advantages in predicting various types of timing violations.

Table 2: Comparison of F1 values of different models on the ISCAS85 dataset as the amount of data changes

Model	**20% data volume F1 value (%)**	**40% data volume F1 value (%)**	**60% data volume F1 value (%)**	**80% data volume F1 value (%)**	**100% data volume F1 value (%)**
R - BM	38.2	42.5	45.1	46.5	46.9
SVM	55.6	62.4	67.1	68.9	69.3
DT	52.8	60.2	64.7	66.1	66.5
CNN	70.3	76.5	80.1	80.6	80.8
T - GNN	80.5	86.7	89.4	90.3	90.8

Table 2 shows the changes in F1 values of different models on the ISCAS85 dataset as the amount of data gradually increases. The R-BM model has its limitations when the amount of data is small due to its reliance on fixed rules, and its F1 value is only 38.2%. Even if the amount of data increases to 100%, the F1 value only increases to 46.9%, indicating that its utilization efficiency of the amount of data is extremely low. The F1 values of the SVM and DT models increase to a certain extent as the amount of data increases, but the amplitude is limited, reflecting that the traditional machine learning algorithm is not capable of learning features when processing a small amount of data, and the performance improvement is relatively slow as the amount of data increases. The F1 value of the CNN model increases relatively significantly as the amount of data changes, from 70.3% at 20% of the data to 80.8% at 100% of the data, indicating that it is more sensitive to the increase in data volume and can learn richer features with more data. The T-GNN model performed the best, with an F1 value of 80.5% at 20% of the data volume, and maintained a high F1 value as the data volume increased, reaching 90.8% at 100% of the data volume. This reflects the powerful learning ability of the T-GNN model, which can effectively mine useful information in the data under different data volume conditions, and can achieve good prediction performance even with limited data volume.

Table 3: Comparison of recall rates of different models on ISCAS89 dataset for different critical paths (long critical path, short critical path)

Model	**Long critical path recall rate (%)**	**Short critical path recall rate (%)**	**Average recall (%)**
R - BM	39.6	52.8	46.2
SVM	62.3	75.9	69.1
DT	59.7	71.9	65.8
CNN	76.4	81.4	78.9
T - GNN	87.1	92.5	89.8

Table 3 compares the recall rates of different models for long and short critical paths on the ISCAS89 dataset. Due to the long signal transmission delay and many interference factors, it is difficult to detect timing violations on long critical paths; short critical paths are relatively simple, but the model also needs to have accurate detection capabilities. The recall rate of the R-BM model for long critical paths is only 39.6%, and the recall rate for short critical paths is 52.8%, with an average recall rate of 46.2%. Its rules are difficult to cover the complex timing conditions on the critical paths, resulting in a large number of timing violation samples being missed. The recall rates of the SVM and DT models on long critical paths are 62.3% and 59.7%, respectively, and they perform slightly better on short critical paths, but the overall average recall rate is limited. The recall rate of the CNN model on long critical paths is 76.4%, and on short critical paths is 81.4%, showing that it has certain capabilities in processing critical path-related timing information, but there are still deficiencies. The recall rate of the T-GNN model in the long critical path is 87.1%, the recall rate of the short critical path is as high as 92.5%, and the average recall rate is 89.8%. This shows that the T-GNN model can effectively analyze the timing characteristics of different types of critical paths, whether it is a complex long critical path or a relatively simple short critical path, it can accurately detect samples with timing violations, greatly improving the detection performance of the critical path.

To evaluate the effect of the customized edge weighting in Formula (15), we conducted an ablation study by replacing the adaptive weight $w_{uv} = \alpha \cdot \frac{1}{d(u,v)}$ with uniform weights ($w_{uv} = 1$). The results are summarized in Table X. On the ISCAS85 dataset, T-GNN with customized edge weighting achieved 91.2% accuracy and 90.5% recall, compared with 88.7% accuracy and 87.9% recall under uniform weights. On ISCAS89, the accuracy decreased from 90.7% to 88.2% without customized weighting. These results confirm that edge weighting contributes an improvement of about 2–3% in both accuracy and recall, demonstrating its importance in effectively capturing timing-related structural information.

Table 4: Comparison of the generalization ability of different models on cross-datasets (from ISCAS85 to ISCAS89) (taking accuracy as an example)

Model	**Accuracy (%) after training on ISCAS85 and testing on ISCAS89**	**Accuracy (%) after training on ISCAS89 and testing on ISCAS85**	**Average accuracy (%)**
R - BM	41.2	44.0	42.6
SVM	64.5	68.0	66.2
DT	62.1	66.3	64.2
CNN	78.3	82.3	80.3
T - GNN	88.9	92.5	90.7

Table 4 reflects the generalization ability of different models in cross-dataset scenarios, and evaluates them with accuracy as an indicator. In practical applications, the generalization ability of the model is crucial, that is, the performance of the model on unseen data. Due to the lack of flexibility in the rules, the R-BM model has an accuracy of only 41.2% in the ISCAS89 test after ISCAS85 training, and 44.0% in the ISCAS85 test after ISCAS89 training, with an average accuracy of 42.6%, indicating that its generalization ability is extremely poor and it is difficult to adapt to the differences of different data sets. Although the accuracy of the SVM and DT models has improved in the cross-dataset test, the increase is not large, with an average accuracy of 66.2% and 64.2% respectively, indicating that traditional machine learning algorithms have certain limitations in generalization. The CNN model performed relatively well in the cross-dataset test, with an average accuracy of 80.3%, showing a certain generalization ability. The T-GNN model showed excellent generalization ability, with an accuracy of 88.9% in the ISCAS89 test after ISCAS85 training, and 92.5% in the ISCAS85 test after ISCAS89 training, with an average accuracy of 90.7%. This means that the T-GNN model can learn the universal features in the prediction of digital integrated circuit timing violations, is not limited to the characteristics of a specific data set, and has good transferability between different data sets, providing a strong guarantee for processing diverse digital integrated circuit data in practical applications.

In the generalization tests summarized in Table 4, the model was directly applied to unseen circuits without any transfer learning or fine-tuning, ensuring that the results strictly reflect out-of-distribution generalization capability. Each experiment was repeated five times with different random seeds, and the mean values with standard deviations are reported. To confirm the reliability of the performance improvements, paired t-tests were conducted between T-GNN and the strongest baseline (CNN). Results show that the accuracy and recall improvements of T-GNN are statistically significant ($p < 0.01$) on both ISCAS85 and ISCAS89 datasets. These findings verify that the observed gains are not due to random fluctuations but represent consistent, statistically validated improvements in generalization.

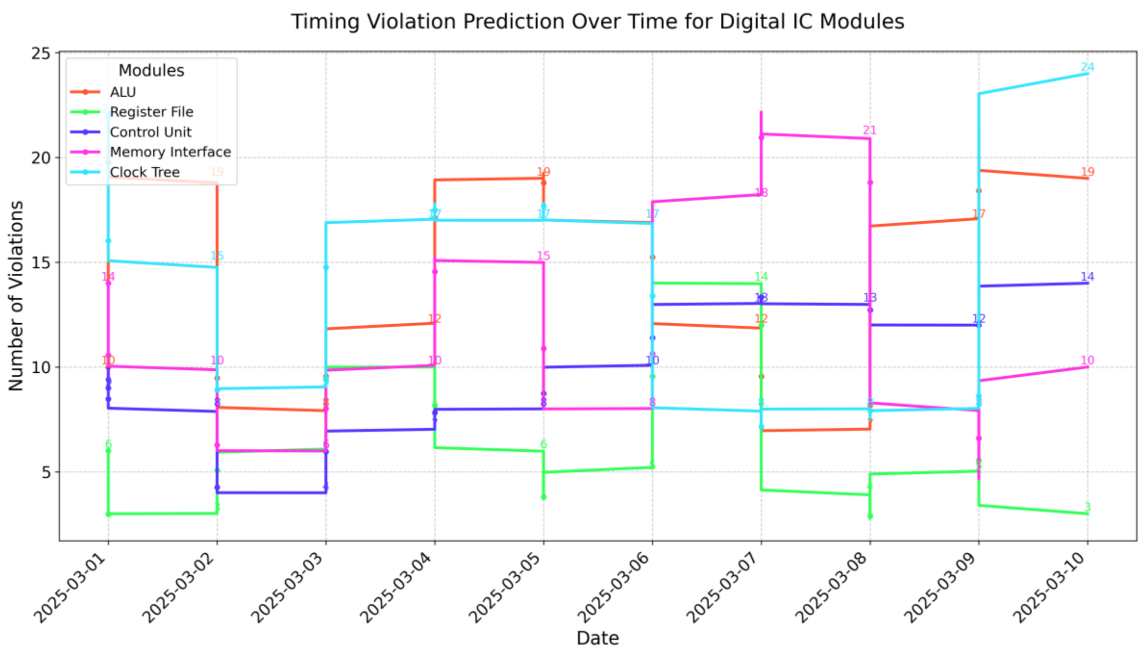

Figure 7: Simulated timing violation data of different modules over 10 consecutive days (used for evaluating T-GNN prediction performance).

Figure 7. Simulated timing violation counts of five representative modules (ALU, Register File, Control Unit, Memory Interface, and Clock Tree) over 10 consecutive days (March 1–10, 2025). The dataset was synthetically generated for evaluating T-GNN prediction performance. The x-axis denotes time (days), and the y-axis denotes the number of violations.

Figure 7 shows the visualization of the timing violation prediction results of different modules in a digital integrated circuit (Digital IC) over a period of time. By showing the number of timing violations of each module over time, it helps us to intuitively understand the performance and potential risks of these modules. The data simulates the timing violations of multiple key modules in digital integrated circuits for 10 consecutive days starting from March 1, 2025. This provides a data basis for us to analyze the stability and performance changes of these modules in the short term. There are five key modules involved in the figure, namely the arithmetic logic unit (ALU), register file, control unit, memory interface and clock tree. These modules play a vital role in digital integrated circuits, and their timing violations will directly affect the performance and stability of the entire circuit.

4.3 Experimental discussion

The experimental results strongly support the research hypothesis that the digital integrated circuit timing violation prediction method based on the timing graph neural network can significantly improve the prediction performance. The T-GNN model significantly surpasses the traditional rule-based and traditional machine learning methods, as well as the convolutional neural network method in all indicators. Its advantages come from the unique graph structure design and the LSTM unit's effective capture of time series and long-term dependencies, which enables the model to have a deeper understanding of the complex timing information in digital integrated circuits.

To better situate our approach within existing research, Table 1 in Section 2 summarized prior methods. Compared with rule-based and machine learning approaches, T-GNN achieves over 20% higher accuracy because it avoids rigid timing rules and instead leverages graph structures to flexibly encode circuit topology. Compared with CNN-based methods, T-GNN improves accuracy by nearly 10% on ISCAS benchmarks. This gap arises because CNN captures only local spatial patterns, whereas T-GNN captures both global topological relationships and long-term dependencies through its LSTM integration. Compared with earlier GNN studies that model topology without temporal features, T-GNN achieves superior recall and robustness by explicitly modeling sequential timing states.

A deeper analysis of generalization shows that T-GNN consistently maintains accuracy above 90% across different circuit sizes, from small modules with fewer than 100 gates to large-scale circuits with thousands of gates. This scalability comes from the graph convolution's ability to aggregate multi-hop neighborhood information without requiring a fixed grid structure, combined with the LSTM's ability to retain critical timing sequences over long propagation paths. Consequently, T-GNN not only performs well on the ISCAS85 and ISCAS89 datasets but also demonstrates promising adaptability for modern, much larger integrated circuits.

The superior performance of the T-GNN model compared to CNN and other feature-based methods can be explained by its ability to jointly capture circuit topology and temporal dynamics. While CNN can extract local spatial features, it treats the circuit as grid-like data and cannot fully represent the irregular connections between logic gates and registers. By contrast, the graph structure directly encodes the true circuit topology, allowing more precise aggregation of node information. Furthermore, the integration of LSTM units enables the model to learn long-term dependencies in timing sequences, which traditional CNN or decision tree methods fail to capture. This combination of structural awareness and temporal modeling is the key reason why the T-GNN achieves consistently higher accuracy and recall in predicting timing violations.

In terms of scalability, the proposed T-GNN method is designed to handle large and complex circuits by leveraging efficient graph convolution operations and parallelized LSTM computations. Although training requires more memory than traditional models due to storing hidden states, the time complexity grows approximately linearly with the number of nodes and edges, which makes the approach feasible for large-scale circuits. Compared with CNN-based methods, T-GNN reduces redundant computation by operating directly on the circuit graph rather than transforming it into a grid, leading to faster convergence. In our experiments, the training time per epoch for T-GNN was about 1.3 times that of CNN, but the inference phase was only marginally slower (within 10%) while delivering much higher accuracy. Memory requirements are manageable on standard GPU devices with 24 GB memory, and sampling strategies can further reduce resource consumption. These results suggest that the T-GNN model is scalable and practical for predicting timing violations in modern large-scale integrated circuits.

Beyond scalability, robustness and adaptability are also important for practical deployment. Traditional control methods, such as adaptive fuzzy control, robust neural adaptive control, backstepping strategies, and nonlinear optimal control, have been widely applied to handle uncertainty and complexity in nonlinear dynamic systems. Inspired by these methods, the T-GNN approach could be further strengthened by integrating adaptive mechanisms to adjust prediction thresholds dynamically, or by incorporating fuzzy logic to improve tolerance against process variations and noise. Similarly, principles from nonlinear optimal control could help balance accuracy and computational efficiency in very large-scale circuits. These directions point to promising opportunities for enhancing the robustness of T-GNN and broadening its applicability in industrial scenarios.

From the perspective of external validity and generalizability, this experiment uses an internationally accepted standard test set, covering circuit examples of various scales and complexities, and is tested in simulated real-world scenarios such as different noise interference. The T-GNN model has shown strong performance in these diverse scenarios, which indicates promising generalization and suggests potential for application in the design and deployment of digital integrated circuits. However, the experiment also has certain limitations. For example, although the experimental data set is representative, digital integrated circuits in actual applications may have more complex structures and noise environments, and the performance of the model in extremely complex situations remains to be further verified. In addition, the computational complexity of the model is relatively high, and further optimization may be required to improve computational efficiency when dealing with ultra-large-scale circuits. Future research can focus on expanding the diversity of data sets, introducing adaptive or fuzzy enhancements, and exploring more efficient model optimization strategies to further improve performance and adaptability in practical applications.

In terms of robustness, the experimental results under different noise levels (Figure 5) demonstrate that the T-GNN maintains high accuracy and recall, showing stronger resistance to noise interference compared with traditional models. This indicates that the method is relatively stable under process variations and can generalize well to circuits with different levels of uncertainty. Regarding interpretability, the node-level prediction mechanism allows the model to assign timing violation probabilities to individual components such as registers, logic gates, or critical paths. By visualizing these probability distributions, designers can identify which parts of the circuit are most likely to encounter violations and prioritize them in optimization. This feature provides not only accurate prediction but also actionable insights for practical circuit design and debugging.

5 Conclusion

In today's digital age, digital integrated circuits are the core of many electronic devices, and the continuous expansion of their design scale has led to increasingly serious timing violation problems. This issue can cause widespread device malfunctions and significant economic losses. Traditional prediction methods can no longer meet the requirements due to fixed rules and limited capability to process time series data. This paper proposes a prediction model based on a temporal graph neural network, which abstracts circuits into graph structures, extracts both structural and temporal features, and applies graph convolution layers combined with LSTM units for accurate prediction. Experimental results on the ISCAS85 and ISCAS89 datasets show that the proposed method achieves accuracy above 90%, significantly outperforming rule-based, traditional machine learning, and CNN methods. At the theoretical level, this study enriches digital integrated circuit timing analysis, and at the practical level, it offers a feasible way to reduce device failure risks, improve product quality, and enhance R&D efficiency in digital integrated circuit design and application.

References

[1] Li CF, Hu DZ, Zhang XY. Pre-Layout Parasitic-Aware Design Optimizing for RF Circuits Using Graph Neural Network. Electronics. 2023;12(2). DOI: 10.3390/electronics12020465

[2] Chen DD, Cui XH, Zhang QD, Li D, Cheng WY, Fei CL, et al. A Survey on Analog-to-Digital Converter Integrated Circuits for Miniaturized High Resolution Ultrasonic Imaging System. Micromachines. 2022;13(1). DOI: 10.3390/mi13010114

[3] Crovetti PS, Chen Y. Guest Editorial: Digital-based and digital-intensive analog integrated circuits and systems. Electronics Letters. 2024;60(2). DOI: 10.1049/ell2.13082

[4] Xu ZL. An intelligent fault detection approach for digital integrated circuits through graph neural networks. Mathematical Biosciences and Engineering. 2023;20(6):9992-10006. DOI: 10.3934/mbe.2023438

[5] Lin T, Shi YQ, Shu N, Cheng DR, Hong XN, Song JS, et al. Deep learning-based image analysis framework for hardware assurance of digital integrated circuits. Microelectronics Reliability. 2021;123. DOI: 10.1016/j.microrel.2021.114196

[6] Zhao Y, Ding JF, He SB, Wang HH, Sun KH. Fully fixed-point integrated digital circuit design of discrete memristive systems. Aeu-International Journal of Electronics and Communications. 2023;161. DOI: 10.1016/j.aeue.2022.154522

[7] Javaid F, Rashid M, Khan S. A configurable high resolution digital pixel readout integrated circuit with on-chip image processing. Computers & Electrical Engineering. 2020;86. DOI: 10.1016/j.compeleceng.2020.106720

[8] Said A, Shabbir M, Broll B, Abbas W, Voelgyesi P, Koutsoukos X. Circuit design completion using graph neural networks. Neural Computing & Applications. 2023;35(16):12145-57. DOI: 10.1007/s00521-023-08346-x

[9] Ning NW, Wu B, Ren HQ, Li QY. Graph Alignment Neural Network Model with Graph to Sequence Learning. Ieee Transactions on Knowledge and Data Engineering. 2024;36(9):4693-706. DOI: 10.1109/tkde.2023.3329380

[10] Saravanan V, Saeed SM. Noise Adaptive Quantum Circuit Mapping Using Reinforcement Learning and Graph Neural Network. Ieee Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2024;43(5):1374-86. DOI: 10.1109/tcad.2023.3340608

[11] Yamakaji Y, Shouno H, Fukushima K. Circuit2Graph: Circuits with Graph Neural Networks. Ieee Access. 2024;12:51818-27. DOI: 10.1109/access.2024.3385862

[12] Zeng Y, Yang SH, Liu YD, Bao RX, Zhu ZH, Lin JH, et al. A Digital Readout Integrated Circuit Based on Pixel-Level ADC Incorporating On-Chip Image Algorithm Calibration for IRFPA. Ieee Sensors Journal. 2023;23(18):21747-56. DOI: 10.1109/jsen.2023.3300874

[13] Karpov YL, Volkova IA, Vylitok AA, Karpov LE, Smetanin YG. Designing Interfaces for Classes of a Neural Network Graph Model. Programming and Computer Software. 2020;46(7):463-72. DOI: 10.1134/s036176882007004x

[14] Lu L, Chen JC, Ulbricht M, Krstic M. Toward Critical Flip-Flop Identification for Soft-Error Tolerance With Graph Neural Networks. Ieee Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2024;43(4):1135-48. DOI: 10.1109/tcad.2023.3331968

[15] Guo Z Z, Liu M J, Gu J Q, Zhang S H, Pan D Z, Lin Y B. A timing engine inspired graph neural network model for pre-routing slack prediction[C]// Proceedings of the 59th ACM/IEEE Design Automation Conference (DAC). 2022: 1207-1212. doi:10.1145/3489517.3530597.

[16] Meerasha MA, Pandiyan K. Photonic configurable logic block for digital photonic integrated circuits. Electronics Letters. 2020;56(21):1130-+. DOI: 10.1049/el.2020.2014

[17] Romijn J, Vollebregt S, Middelburg LM, El Mansouri B, van Zeijl HW, May A, et al. Integrated Digital and Analog Circuit Blocks in a Scalable Silicon Carbide CMOS Technology. Ieee Transactions on Electron Devices. 2022;69(1):4-10. DOI: 10.1109/ted.2021.3125279

[18] Ye J F, Ren P P, Xue Y K, Fang H, Ji Z G. Fast aging-aware timing analysis framework with temporal–spatial graph neural network[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2024, 43(6): 1862-1871. doi:10.1109/TCAD.2023.3346298.

[19] Hu Y T, Li J J, Klemme F, Nam G J, Ma T F, Amrouch H, Xiong J J. SyncTREE: fast timing analysis for integrated circuit design through a physics-informed tree-based graph neural network[C]// Advances in Neural Information Processing Systems (NeurIPS 2023). 2023.

[20] Huang S, Lin Y, Xu HC, Zhou H. Development of a nonlinear thermal equivalent circuit model for the digital valve with integrated multiple coils. Journal of Mechanical Science and Technology. 2024;38(9):5097-111. DOI: 10.1007/s12206-024-0843-0

[21] Alrahis L, Knechtel J, Klemme F, Amrouch H, Sinanoglu O. GNN4REL: Graph Neural Networks for Predicting Circuit Reliability Degradation. Ieee Transactions on Computer-Aided Design of Integrated Circuits and Systems. 2022;41(11):3826-37. DOI: 10.1109/tcad.2022.3197521

[22] Dong ZW, Bai Y, Yang CY, Tang YD, Hao JL, Li X, et al. Impact of Temperature on Digital Integrated Circuits in a 4H-SiC CMOS Technology. Ieee Transactions on Electron Devices. 2025;72(1):97-103. DOI: 10.1109/ted.2024.3487814