Deep H-GCN: Fast Analog IC Aging-Induced Degradation Estimation

Tinghuan Chen, *Graduate Student Member, IEEE*, Qi Sun, *Graduate Student Member, IEEE*, Canhui Zhan, Changze Liu, *Member, IEEE*, Huatao Yu, and Bei Yu, *Member, IEEE*

***Abstract*—With continued scaling, the transistor aging induced by hot carrier injection (HCI) and bias temperature instability (BTI) causes an increasing failure of nanometer-scale integrated circuits (ICs). Compared to digital ICs, analog ICs are more susceptible to aging effects. The industrial large-scale analog ICs bring grand challenges in the efficiency of aging verification. In this article, we propose a heterogeneous graph convolutional network (H-GCN) to fast estimate aging-induced transistor degradation in analog ICs. To characterize the multityped devices and connection pins, a heterogeneous directed multigraph is adopted to efficiently represent the topology of analog ICs. A latent space mapping method is used to transform the feature vector of all typed devices into a unified latent space. We further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings. The extended deep H-GCN can extract information from multihop devices without an oversmoothing issue. A probability-based neighborhood sampling method on the bipartite graph is adopted to ease the model training on large-scale graphs and achieve good scalability. Experiments on very advanced 5-nm industrial benchmarks show that, compared to traditional graph learning methods and static aging reliability simulations by an industrial design-for-reliability (DFR) tool, the proposed deep H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared to the dynamic and static aging reliability simulations, our extended deep H-GCN, on average, can achieve 241× and 39× speedup, respectively.**

***Index Terms*—Aging, analog integrated circuits, bias temperature instability (BTI), degradation, graph convolutional networks, hot carrier injection (HCI).**

I. Introduction

With continued scaling, the susceptibility of nanometer-scale transistors to aging-related wear-out phenomena has increased significantly in integrated circuits (ICs) [1]. As illustrated in Fig. 1, two primary aging-related wear-out mechanisms of semiconductor-based micro-electronic devices are: 1) bias temperature instability (BTI) and 2) hot carrier injection (HCI) [2]. BTI mechanism is that accumulated holes in silicon/oxide interface result in breaking of Si-H bonds as shown in Fig. 1(a). The HCI mechanism is that due to high electric field in the drain side, hot carriers cause breaking of the Si-H bond and traps the oxide bulk as shown in Fig. 1(b) [3]. These aging effects cause transistor parameters, e.g., threshold voltage, to shift from their nominal values over time, resulting in a gradual circuit failure. Although there are some advanced systematic calibrations, it is hard to achieve perfect reconstruction [4], [5]. Compared to digital ICs, analog ICs are more susceptible to these transistor parameters.

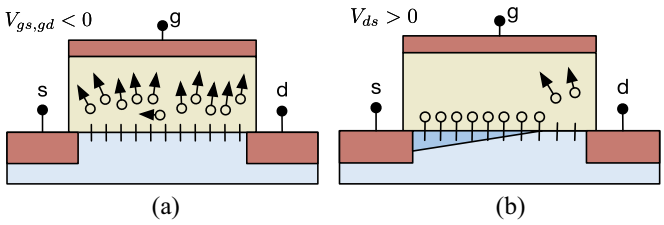

Fig. 1. BTI and HCI mechanisms. (a) BTI: Accumulated holes in silicon/oxide interface result in breaking of Si-H bonds. (b) HCI: Due to high electric field in drain side, hot carriers cause breaking of Si-H bond and traps oxide bulk.

In order to save development costs and provide the opportunity for interactive feedback during the design process, estimating aging-induced transistor degradation before committing the design to silicon is a key step. In the industry, according to the estimated aging-induced transistor degradation, the aging violations can be judged. If there are one or more aging violations in the aging verification, as a typical verification-then-fix approach, design-for-reliability (DFR) designers will go back to the design stages to fix them. Besides, compared to prelayout netlists, post-layout netlists contain parasitic capacitances and resistances. Thus, the aging reliability simulation on post-layout netlists has more accurate judgment. However, post-layout netlists have a larger scale size than prelayout netlists so that they bring grand challenges in the efficiency of aging verification.

In the industry, as shown in Fig. 2, a traditional aging reliability simulator is adopted to verify the circuit reliability. The typical aging reliability simulation is classified into static simulation and dynamic simulation [6]. They both take a circuit netlist and its stress conditions (stimuli) as inputs. A fresh simulation is performed to obtain the fresh transistor parameters. According to the fresh transistor parameters, the stress simulation is performed to get device degradations. By using the device degradations, the aging simulation is adopted to

Manuscript received May 22, 2021; revised July 5, 2021; accepted August 2, 2021. Date of publication August 24, 2021; date of current version June 20, 2022. This work was supported in part by the HiSilicon Technologies Company, and in part by the Research Grants Council of Hong Kong SAR under Project 14209420. A preliminary version has been presented at the ACM/IEEE Asia and South Pacific Design Automation Conference (ASPDAC) in 2021. This article was recommended by Associate Editor S. Mohanty. *(Corresponding author: Bei Yu.)*

Tinghuan Chen, Qi Sun, and Bei Yu are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: byu@cse.cuhk.edu.hk).

Canhui Zhan, Changze Liu, and Huatao Yu are with the COT Department, HiSilicon Technologies Company, Shenzhen 518129, China.

Digital Object Identifier 10.1109/TCAD.2021.3107250

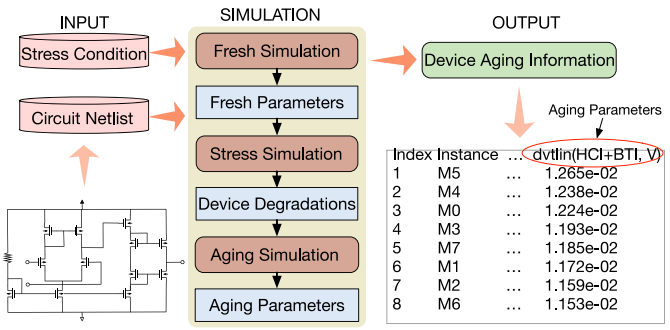

Fig. 2. Traditional aging reliability simulation.

obtain the aging transistor parameters. Finally, the shift values from fresh transistor parameters to the aging transistor parameters are output to judge the circuit reliability. Compared to the static simulation, the traditional dynamic simulation considers dynamic stress conditions, such as clock speeds in the fresh simulation and the aging simulation stages so that it needs a large number of accepted transient steps. Thus, the dynamic aging reliability simulation can obtain a more accurate aging-induced transistor degradation while takes more time. Nevertheless, the accuracy of traditional dynamic aging reliability simulation heavily relies on the given dynamic stress conditions.

Modeling aging-induced transistor degradation has been widely studied in the literature. Tu *et al.* [7] adopted a voltage-controlled current source model to predict HCI issues in the design stage. A transistor drain-current surrogate model was developed to explore the effects of degradation on analog circuits [8]. Other analytical models were also surveyed and concluded in [3]. However, there are several limitations and drawbacks to these analytical models. First, a correct judgment on the reliability of the circuits heavily relies on the appropriate stress conditions [9]. The static aging reliability simulation causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. Second, it is time consuming to achieve accurate detections since the dynamic aging simulation needs a large number of accepted transient steps. It is usually difficult to find a compromise between computational complexity and model accuracy.

Convolutional neural networks (CNNs), as a data-driven approach, have achieved great successes in computer version [10], [11] and circuit design [12]–[14]. Typically, the aging reliability simulations are performed on circuit netlists. Intuitively, an analog IC netlist can be naturally represented as a graph. However, the graph is irregular grid-based data, which is not as straightforward as the convolution and pooling in traditional CNNs. Graph convolutional networks (GCNs) were proposed to perform machine learning tasks on these irregular grid-based data [15]–[17]. Recently, GCNs were adopted to predict observation point candidates on the design-for-testability [18], select timing models [17], estimate layout parasitics [19], annotate netlists [20], [21], guide placement [22], and tune transistor sizing [23]. However, the analog IC netlists have heterogeneity since typical analog ICs contain multityped basic devices (e.g., transistor and resistor) and multityped connection pins (e.g., drain and gate). Typically, the existing GCNs adopt one-hot encoding to distinguish among multityped basic devices and connection pins. However, it will miss some structural information among multityped nodes as well as unstructured content associated with each node [24].

In our previous work [25], considering that the multityped devices and connection pins in analog ICs, we propose a heterogeneous GCN (H-GCN) to fast and accurately estimate aging-induced transistor degradation. Besides, a probability-based neighborhood sampling algorithm on the heterogeneous graph is proposed to facilitate GPU training. Despite H-GCN can achieve enormous successes in the estimation of aging-induced transistor degradation, it is very shallow [25]. Such a shallow model limits its ability to extract information from multihop devices. Simply stacking more layers and adding nonlinearity may cause performance degradation, due to the oversmoothing phenomenon [26]. Besides, the proposed sampling algorithm on the heterogeneous graph in [25] has high computational complexity for the large-scale analog IC netlist since the runtime is dominated by graph transformation.

Very recently, several arts try to tackle the over-smoothing issue. Jumping knowledge networks (JKNets) combine the output of each layer to the last layer to keep the local properties of the node representations [27]. A few edges are randomly removed from the input graph in DropEdge to alleviate the oversmoothing issue [28]. Simplifying GCNs captures higher-order features in the graph by using the kth power of the graph convolution in each neural network layer [29]. Personalized PageRank is generalized to an arbitrary graph diffusion process in graph diffusion convolution [30], [31]. GCNs with initial residual connections and identity mappings (GCNII) ensure that the final representation of each node retains at least a fraction from the input layer, even if many layers are stacked [32]. Nevertheless, these models do not consider the heterogeneity of the graph.

In this article, we make the following contributions.

1) Considering multityped connection pins and devices with different design parameters, we adopt a heterogeneous directed multigraph to represent a post-layout netlist efficiently.
2) We propose an H-GCN to fast estimate the aging-induced transistor degradation. In our proposed H-GCN, an embedding generation algorithm with a latent space mapping method is developed to aggregate information from the node itself and its multityped neighboring nodes through multityped edges.
3) We further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings. The extended deep H-GCN can extract information from multihop devices without an oversmoothing issue.
4) A neighborhood sampling method is used to ease large-scale graph training and reduce GPU memory overhead. The neighborhood sampling method is performed on the bipartite graph, which is constructed by directly parsing and flattening the circuit netlist. Therefore, our proposed method has enough scalability to handle the large-scale analog ICs.

5) We conduct experiments on advanced 5-nm industrial designs to show that the proposed deep H-GCN can achieve faster and more accurate estimations of aging-induced transistor degradation, compared with the traditional graph learning methods and static aging reliability simulations by an industrial DFR tool.

The remainder of this article is organized as follows. In Section II, we give our problem formulation and preliminaries about the graph representation of analog ICs, and GCNs. In Section III, we present a heterogeneous directed multigraph to represent the analog IC topology. In Section IV, we propose an H-GCN model, where a novel latent space mapping method is described in detail. In Section V, we further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings. In Section VI, we propose a neighborhood sampling on the bipartite graph to achieve good scalability. In Section VII, we broadly introduce our proposed whole flow. Section VIII presents experimental results, followed by the conclusion in Section IX.

II. Preliminaries

A. Problem Formulation

Before committing the design to silicon, the aging-induced transistor degradation needs to be accurately estimated in the post-layout simulation to judge circuit reliability. However, the actual degree of the aging-induced transistor degradation is hard to estimate. On the one hand, the traditional static aging reliability simulation causes inaccurate judgment on the aging-prone transistors since the dynamic stress conditions are completely ignored. On the other hand, the traditional dynamic simulation is time consuming since it needs a large number of accepted transient steps. Besides, the accuracy of traditional dynamic aging reliability simulation heavily relies on dynamic stress conditions, such as clock speeds and waveform swings. Thus, the traditional aging reliability simulation is hard to make a better tradeoff between accuracy and runtime.

In this work, we adopt a data-driven approach in a supervised-learning manner to fast estimate aging-induced degradation. As mentioned above, the accuracy of traditional dynamic aging reliability simulation heavily relies on dynamic stress conditions. Thus, our data-driven approach decouples dynamic stress conditions. In the training set, in order to achieve accurate estimations, the golden-truth aging-induced degradations are obtained by an industrial aging DFR tool with the static and dynamic stress conditions given by very sophisticated designers. Compared to static stress conditions, it is hard to determine dynamic stress conditions in practice; thus, in the inference stage, dynamic stress conditions are not considered in our data-driven approach. This strategy can achieve a better compromise between computational complexity and model accuracy for the model implementation. In the industry, `dvtlin (HCI+BTI,10)` is used to assess the degree of aging-induced transistor degradation [33], [34]. `dvtlin(HCI+BTI,10)` is defined as follows.

Definition 1 [`dvtlin(HCI+BTI,10)`]: The shifting value of threshold voltage of the transistor from fresh to ten years due to HCI and BTI.

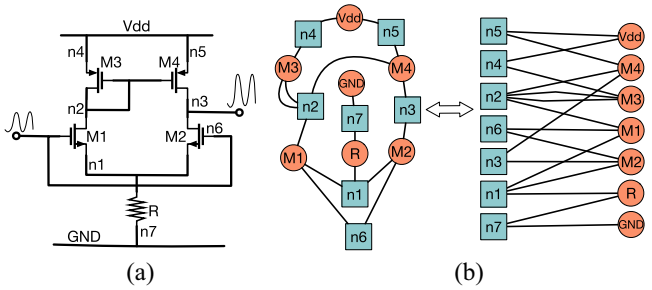

Fig. 3. Analog circuit is represented as a bipartite graph. (a) Analog circuit. (b) Corresponding bipartite representation.

The larger `dvtlin (HCI+BTI,10)` is, the worse aging-induced transistor degradation is, vice versa. For convenience, in this article, we shorten `dvtlin(HCI+BTI,10)` as `dvtlin`.

Based on the above description, we define our problem formulation as follows.

Problem 1 (Estimating `dvtlin` in Analog ICs): Given some analog IC post-layout netlists, their stress conditions and a list containing all transistors with `dvtlin` obtained by an industrial DFR tool as the training set, our task is training a model on the training set to fast and accurately estimate `dvtlin` of each transistor on the testing set while minimizing the estimation error.

B. Analog ICs Topology

When analog IC netlist files are parsed and flattened, analog ICs are naturally represented as bipartite graphs [35]. There are two disjoint sets in the bipartite graph. One set contains all nets and the other one contains all devices. Each undirected edge connects one net and one device. Fig. 3 demonstrates an analog circuit and the corresponding bipartite graph representation.

C. Graph Convolutional Networks

Since the graph can reveal structural information, graph learning provides a powerful data-driven approach for modeling irregular grid-based data in machine learning tasks [15], [16]. Node embedding generation is one of the most critical technologies in graph learning [36]. The representation of each node is obtained by embedding and then fed into the traditional machine learning models for estimation and prediction. Recently, some data-driven methods were proposed to learn node embeddings automatically [15], [16].

One of the data-driven embedding methods is GCN, whose main idea is to automatically learn how to aggregate information iteratively from neighborhoods on graphs using neural networks. The embedding generation is equivalent to using a filter on each node, whose accessible domain contains the node itself and its neighboring nodes. The neighboring nodes are a set of all nodes that are adjacent to the node itself. Compared to traditional CNNs, the graph topology is also treated as a feature to perform machine learning tasks so that GCNs can achieve higher accuracy on graph-based datasets and applications.

III. Heterogeneous Graph Representation

To adopt a GCN-based framework, it is fundamental to perform an embedding generation algorithm on graphs to encode the nodes and edges as feature vectors. Although the graph representations of analog ICs have been studied in many works [35], [37], [38], the embedding generation algorithm cannot be directly performed on the graph representations of analog ICs since not all nodes have design parameters as features. Next, we will propose a directed heterogeneous graph representation method to guarantee that each node has features.

To perform an embedding generation algorithm, we treat each device, direct current (dc) voltage source or ground as a node. An analog IC post-layout netlist can be flattened as a bipartite graph as illustrated in Fig. 3. To guarantee that each node has features, a naïve method is constructing a homogeneous multigraph to represent the topology of analog IC netlist [22]. In this homogeneous multigraph, each edge represents one path from one device to another. It allows multiple edges between any two nodes since there may be multiple paths from one device to the other one. Note that dynamic stress conditions are not to be considered in this graph representation so that the GCN-based framework is independent of them and can replace the traditional static aging reliability simulation.

The typical analog IC netlists have heterogeneity since they contain multityped basic devices and multityped connection pins [39]. An inevitable problem of the homogeneous representation method is that it fails to characterize the diversities among pins, devices, connections, and relative sequential relationships. In order to express these diversities, we propose a heterogeneous directed multigraph representation, where the type of the edges is treated as the type of the pins to which it connects. The heterogeneous directed multigraph is defined as follows.

Definition 2 (Heterogeneous Directed Multigraph): A heterogeneous directed multigraph is defined as a graph $\mathcal{HMG}_d(\mathcal{V}_{\text{hmg}}, \mathcal{E}_{\text{hmg}}, \mathcal{O}_{\mathcal{V}_{\text{hmg}}}, \mathcal{R}_{\mathcal{E}_{\text{hmg}}}, \varphi_{\text{hmg}})$, where $\mathcal{V}_{\text{hmg}}$ is the set of nodes and $\mathcal{E}_{\text{hmg}}$ is the multiset of edges. $\mathcal{O}_{\mathcal{V}_{\text{hmg}}}$ and $\mathcal{R}_{\mathcal{E}_{\text{hmg}}}$ represent the sets of node types and edge types, respectively. φ_{hmg} is adopted to assign each node in $\mathcal{V}_{\text{hmg}}$ to a node type, i.e., $\varphi_{\text{hmg}}(\upsilon_i) \in \mathcal{O}_{\mathcal{V}_{\text{hmg}}}$ for $\forall \upsilon_i \in \mathcal{V}_{\text{hmg}}$. r indicates the edge type, such that $r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}}$. Each instance in $\mathcal{E}_{\text{hmg}}$ is $((\upsilon_i, \upsilon_j), r)$ and the ordered pair (υ_i, υ_j) satisfies $\upsilon_i, \upsilon_j \in \mathcal{V}_{\text{hmg}}$.

We use an example to show this heterogeneous directed multigraph representation. As shown in Fig. 4(a), if we stand at the gate of the transistor `M3` and look out, we will see transistors `M1`, `M3`, and `M4`. Thus, there are three directed edges (denoted by dark green) with gate connections from `M1`, `M3`, and `M4` to the transistor `M3` as shown in Fig. 4(b). This method is inspired by circuit analysis, where the input impedance is obtained in the same fashion [39]. In the same manner, we can obtain the directed multigraph corresponding to the gate connections as shown in Fig. 4(b), where all green edges denote the gate connection. The circuit netlist in Fig. 4(a) is finally transformed into the multigraph in Fig. 5. In particular, considering that the electrical characteristics of dc voltage sources and grounds are not influenced by other devices, we set them to be predecessors.

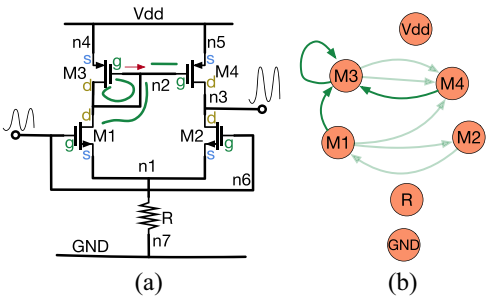

Fig. 4. Heterogeneous directed multigraph representation. (a) Differential amplifier. (b) Gate connection.

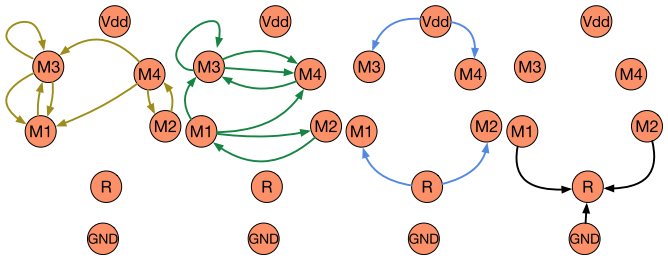

Fig. 5. Heterogeneous directed multigraph with multityped edges. From left to right: drain, gate, source and other connections.

After the topology of different connections is obtained, we use one adjacency matrix to encode the topology of each connection in the heterogeneous multigraph. The adjacency matrix of the type-r connection is defined as $\boldsymbol{A}_r$, where each element $A_r(i, j)$ is the number of the instance $((\upsilon_i, \upsilon_j), r)$ in the multiset $\mathcal{E}_{dmg}$ and $r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}}$. As an example, the adjacency matrix $\boldsymbol{A}_g$ corresponding to the gate connection in Fig. 4(b) is shown in Matrix (1) as follows:

$$\begin{array}{c} \\ M1 \\ M2 \\ M3 \\ M4 \\ R \\ Vdd \\ GND \end{array} \begin{array}{c} \begin{array}{ccccccc} M1 & M2 & M3 & M4 & R & Vdd & GND \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}. \quad (1)$$

In summary, an analog IC netlist can be represented as a heterogeneous directed multigraph. The topology of the heterogeneous directed multigraph is encoded as multiple adjacency matrices, where each nonzero value encodes one type of connection. In the next section, we will propose an H-GCN model on the heterogeneous directed multigraph to estimate `dvtlin` by taking multiple adjacency matrices and the design parameters of all typed devices as inputs.

IV. Heterogeneous GCN

In this section, a H-GCN model on the heterogeneous directed multigraph is developed to estimate `dvtlin`. The features of all types of devices are first mapped into a unified

latent space and then our proposed H-GCN model sequentially conducts multiple embedding generations on these features.

A. Notations

Given an analog IC netlist, we parse and flatten it to be a bipartite graph as shown in Fig. 3. Then, we transform to be a heterogeneous directed multigraph $\mathcal{HMG}_d(\mathcal{V}_{\text{hmg}}, \mathcal{E}_{\text{hmg}}, \mathcal{O}_{\mathcal{V}_{\text{hmg}}}, \mathcal{R}_{\mathcal{E}_{\text{hmg}}}, \varphi_{\text{hmg}})$ as shown in Figs. 4 and 5. Then, according to the heterogeneous directed multigraph, we can obtain $|\mathcal{R}_{\mathcal{E}_{\text{hmg}}}|$ adjacency matrices, i.e., $\boldsymbol{A}_r$ for $\forall r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}}$, where $|\cdot|$ is the cardinality of set. Besides, the device $\upsilon_i \in \mathcal{V}_{\text{hmg}}$ has $h_{\varphi_{\text{hmg}}(\upsilon_i)}$ design parameters for $\varphi_{\text{hmg}}(\upsilon_i) \in \mathcal{O}_{\mathcal{V}_{\text{hmg}}}$. We use these design parameters as the features of each node, i.e., $\boldsymbol{x}_{\upsilon_i} \in \mathbb{R}^{1\times h_{\varphi_{\text{hmg}}(\upsilon_i)}}$. Note that the number of design parameters relies on the type of device. Next, we will propose a unified latent space mapping method to map the features of each node into a unified latent space so that the embedding generation can be performed; meanwhile, the features of multityped devices can be well effectively exploited and encoded.

B. Unified Latent Space Mapping

A typical analog IC netlist contains multityped devices, which have different design parameters. We regard each device as a node and use its design parameters as the node feature vector. To perform an embedding generation algorithm and aggregate information from the node itself and its multityped neighboring nodes, a straightforward method is concatenating all of these design parameters as a long feature vector with one-hot encoding [22]. However, it will miss some structural information among multityped nodes as well as unstructured content associated with each node [24]. In this article, a latent space mapping method is used to transform the feature vectors of all types of nodes into a unified latent space.

We propose to use $|\mathcal{O}_{\mathcal{V}_{\text{hmg}}}|$ node-type-related matrices to map the original feature vectors. For a node $\upsilon_i \in \mathcal{V}_{\text{hmg}}$ whose node type is $t = \varphi_{\text{hmg}}(\upsilon_i) \in \mathcal{O}_{\mathcal{V}_{\text{hmg}}}$ and feature vector is $\boldsymbol{x}_{\upsilon_i} \in \mathbb{R}^{1\times h_t}$, we define a node-typed-related matrix $\boldsymbol{U}_t \in \mathbb{R}^{h_t\times\tau}$ to map the feature vector with length h_t into a unified τ-dimension latent space, i.e., $\boldsymbol{f}_{\upsilon_i}^{(0)} = \boldsymbol{x}_{\upsilon_i} \cdot \boldsymbol{U}_t \in \mathbb{R}^{1\times\tau}$. In order to ease the model training on GPUs, we extend it to be a matrix–matrix multiplication as follows:

$$\boldsymbol{F}^{(0)} = \sum_{t\in\mathcal{O}_{\mathcal{V}_{\text{hmg}}}} \boldsymbol{x}_t \cdot \boldsymbol{U}_t \in \mathbb{R}^{|\mathcal{V}_{\text{hmg}}|\times\tau} \tag{2}$$

where $\boldsymbol{x}_t \in \mathbb{R}^{|\mathcal{V}_{\text{hmg}}|\times h_t}$ is a feature matrix stacking the feature vectors of all type-t nodes. In particular, a node feature vector is $\boldsymbol{0} \in \mathbb{R}^{1\times h_t}$ if the type of the node is not t. $\boldsymbol{F}^{(0)}$ is the feature matrix, where the feature vectors of all nodes are in a unified latent space. We take the circuit in Fig. 4(a) as an example. There are three types of nodes (devices), i.e., $\mathcal{O}_{\mathcal{V}_{\text{hmg}}} = \{vs, trans, res\}$. Thus, there are three feature matrices: $\boldsymbol{X}_{vs} = [\boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{x}_{Vdd}^\top, \boldsymbol{x}_{GND}^\top]^\top$; $\boldsymbol{X}_{\text{trans}} = [\boldsymbol{x}_{M1}^\top, \boldsymbol{x}_{M2}^\top, \boldsymbol{x}_{M3}^\top, \boldsymbol{x}_{M4}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top]^\top$; and $\boldsymbol{X}_{\text{res}} = [\boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top, \boldsymbol{x}_R^\top, \boldsymbol{0}^\top, \boldsymbol{0}^\top]^\top$.

It is noted that the concatenated feature representation [22] is a special case of latent space representation. Compared to the concatenated features representation, our latent space representation is general enough and can effectively exploit and encode features. Furthermore, the proposed latent space mapping method can be extended to multiple layers in the same fashion with a multilayer perceptron (MLP) to extract high-order features.

C. Embedding Generation

After applying the proposed latent space mapping method, the features of all nodes share the same representation forms. An H-GCN model is proposed as the skeleton of the estimation model. The model takes the unified latent feature representations of all nodes $\boldsymbol{F}^{(0)}$ and $|\mathcal{R}_{\mathcal{E}_{\text{hmg}}}|$ adjacency matrices $\boldsymbol{A}_r$ ($r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}}$) as inputs.

As discussed in Section III, we use one adjacency matrix to represent one type of connection in the heterogeneous directed multigraph. In order to distinguish among different connection pins, inspired by [18], we assign a learnable model coefficient to each adjacency matrix. Then, the overall topology of the heterogeneous directed multigraph can be expressed as the summation of the adjacency matrices of all types of connections with these model coefficients. For example, as shown in Fig. 5, there are four adjacency matrices $\boldsymbol{A}_g$, $\boldsymbol{A}_s$, $\boldsymbol{A}_d$, and $\boldsymbol{A}_{otr}$ in the heterogeneous directed multigraph. We assign four model coefficients w_g, w_s, w_d, and w_{otr} to distinguish them. To improve numerical stability, we normalize all adjacency matrices. Therefore, the topology of the heterogeneous directed multigraph can be encoded as $w_g\tilde{\boldsymbol{A}}_g + w_s\tilde{\boldsymbol{A}}_s + w_d\tilde{\boldsymbol{A}}_d + w_{otr}\tilde{\boldsymbol{A}}_{otr}$, where the normalized adjacency matrix $\tilde{\boldsymbol{A}}_r = \boldsymbol{A}_r\boldsymbol{D}_r^{-1}$, and $\boldsymbol{D}_r$ is a diagonal matrix with $D_r(i,i) = \sum_{j=1}^{|\mathcal{V}_{\text{hmg}}|} A_r(i,j)$ and $r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}} = \{g, s, d, otr\}$. For the sake of convenience, we denote the heterogeneous adjacency matrix as follows:

$$\tilde{\boldsymbol{A}} \triangleq \sum_{r\in\mathcal{R}_{\mathcal{E}_{\text{hmg}}}} w_r\tilde{\boldsymbol{A}}_r \tag{3}$$

where w_r is a model coefficient. $\tilde{\boldsymbol{A}}_r$ is the normalized adjacency matrix. r denotes the connection type.

Typically, there are two popular GCN models. One is vanilla GCN [15], where the information of the node itself and that of its neighboring nodes are aggregated by adding one identity matrix to the normalized graph adjacency matrix. The other is GraphSAGE [16], where the feature of the node itself is concatenated with that of its neighboring nodes. There are self-loops in our graph representation as shown in Figs. 4 and 5. Compared with the vanilla GCN, GraphSAGE has stronger ability to distinguish between the self-loop edge and the node itself. Thus, we adopt GraphSAGE as our basic backbone. In our proposed H-GCN, the developed heterogeneous embedding generation is shown as follows:

$$\boldsymbol{F}^{(l)} = \sigma\left(\text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\right) \cdot \boldsymbol{W}^{(l)}\right) \tag{4}$$

where $\text{CONCAT}(\cdot)$ denotes the concatenation operation. $\sigma(\cdot)$ is a nonlinear activation function. In this article, we use ReLU [40] as our nonlinear activation function. $\boldsymbol{W}^{(l)}$ denotes the learnable model coefficients in the lth embedding generation layer.

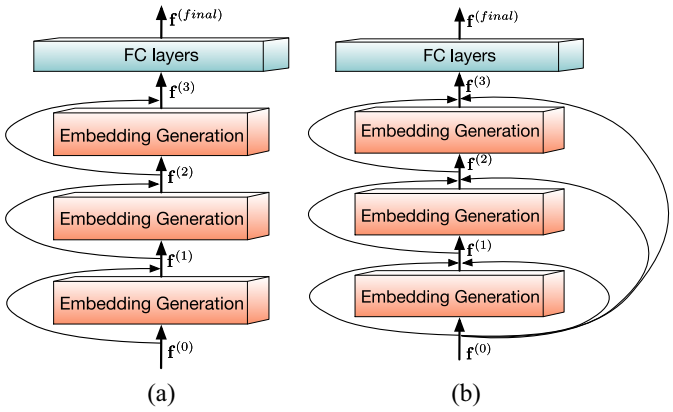

Fig. 6. (a) H-GCN. (b) Deep H-GCN with initial residual connections.

The proposed H-GCN is shown in Fig. 6(a), where the embedding generation as shown in (4) is recursively and sequentially performed several times, the local topology and node features are extracted to represent the feature vector of each node. Then, the extracted feature vector of each node is fed into a traditional machine learning model to estimate `dvtlin`.

V. Going to Deep H-GCN

Despite H-GCN can achieve enormous successes in the estimation of aging-induced transistor degradation, it is very shallow. Such a shallow model limits its ability to extract information from multihop devices so that it cannot achieve a more accurate estimation for complex and large-scale circuit netlists. Besides, stacking more layers and adding nonlinearity tend to degrade the performance of this model, due to the oversmoothing issue [26]. In this section, based on GCNII [32], we further extend the proposed H-GCN to be a deep version via initial residual connections and identity mappings to alleviate the oversmoothing issue.

A. Embedding Generation With Initial Residual Connections and Identity Mappings

The primary reason behind the oversmoothing issue is that the representations of the nodes as shown in (4) become indistinguishable as the number of heterogeneous embedding generation layers increases. In order to alleviate the oversmoothing issue, according to GCNII [32], the final representation of each node retains the input layer even if many layers are stacked. According to this manner, we extend the heterogeneous embedding generation as shown in (4) by concatenating with the input of the first layer as follows:

$$\boldsymbol{F}^{(l)} = \sigma\left(\text{CONCAT}\left(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}\right) \cdot \boldsymbol{W}^{(l)}\right) \quad (5)$$

where $\boldsymbol{F}^{(0)}$ is the unified feature matrix defined in (2). The heterogeneous adjacency matrix $\tilde{\boldsymbol{A}}$ is defined in (3). As shown in Fig. 6(b), our extended deep H-GCN leverages the embedding generation with the initial residual connections. Thus, the final representation of each node retains the input layer even if many layers are stacked.

Algorithm 1 Embedding Generation of Deep H-GCN

Require: $|\mathcal{R}_{\mathcal{E}_{hmg}}|$ normalized adjacency matrices $\tilde{\boldsymbol{A}}_r$ and connection-type-related model coefficients w_r with $\forall r \in \mathcal{R}_{\mathcal{E}_{hmg}}$; feature matrices of all nodes corresponding to each type of nodes $\boldsymbol{X}_t$ and node-type-related latent space mapping matrices $\boldsymbol{U}_t$ with $\forall t \in \mathcal{O}_{\mathcal{V}_{hmg}}$; Search depth D; Model coefficients matrices $\boldsymbol{W}^{(l)}$ for $l = 1, 2, \cdots, D$.

$\boldsymbol{F}^{(0)} \leftarrow \sum_{t \in \mathcal{O}_{\mathcal{V}_{hmg}}} \boldsymbol{X}_t \boldsymbol{U}_t$; ▷ map to a unified latent space

$\tilde{\boldsymbol{A}} \leftarrow \sum_{r \in \mathcal{R}_{\mathcal{E}_{hmg}}} w_r \tilde{\boldsymbol{A}}_r$; ▷ encode the topology of the heterogeneous graph

for $l = 1$ to D **do**

  $\beta_l \leftarrow \log(1/l + 1)$;

  $\boldsymbol{F}^{(l)} \leftarrow \sigma(\text{CONCAT}(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}) \cdot ((1 - \beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)})$; ▷ embedding generation

end for

return Embedding feature matrix $\boldsymbol{F}^{(D)}$.

According to [30] and [32], performing multiple nonlinearity operations to the feature matrix still causes the oversmoothing issue. In our deep H-GCN model, inspired by ResNet [41], to further alleviate the oversmoothing issue, we leverage an identity matrix $\boldsymbol{I}$ with a hyperparameter β_l to the model coefficient matrix $\boldsymbol{W}^{(l)}$ in (5) as follows:

$$\boldsymbol{W}^{(l)} \leftarrow (1 - \beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)}. \quad (6)$$

Note that $\boldsymbol{I}$ is easy to extend as an augmented matrix of an identity matrix with a zero matrix if $\boldsymbol{W}^{(l)}$ is not a square matrix. By adding the matrix $\boldsymbol{I}$, we have two advantages: 1) similar to the motivation of ResNet [41], the matrix $\boldsymbol{I}$ ensures that a deep H-GCN model achieves the same performance as its shallow version does and 2) since the optimal model coefficient matrix $\boldsymbol{W}^{(l)}$ has small norm, it allows to put strong regularization. Chen *et al.* suggested $\beta_l = \log(1/l + 1)$ [32].

Like our proposed H-GCN, our extended deep H-GCN recursively and sequentially performs the embedding generation with initial residual connections and identity mappings several times to extract the information of the local topology and node features to represent the feature vector of each node. Then, the extracted feature vector is fed into a traditional machine learning model to estimate `dvtlin` of each transistor.

Unlike JKNet [27], our extended deep H-GCN can facilitate several layers stacking since the output of each layer does not be uniformly combined to the last layer. Compared with GCNII [32], our extended deep H-GCN has the stronger ability to distinguish between the self-loop edge and the node itself since it concatenates the features of neighboring nodes and the node itself. By using the extended deep H-GCN, it is expected to achieve more accurate estimations of aging-induced transistor degradation.

Our proposed deep embedding generation algorithm (i.e., forward propagation) is summarized in Algorithm 1. The connection-type-related normalized adjacency matrices $\tilde{\boldsymbol{A}}_r$, connection-type-related model coefficients w_r, feature matrices of all nodes $\boldsymbol{X}_t$, node-type-related latent space mapping matrices $\boldsymbol{U}_t$, the search depth D, and model coefficients matrices

$\boldsymbol{W}^{(l)}$ are provided as inputs. According to (2), the latent space mapping matrices $\boldsymbol{U}_t$ are used to map features of all nodes $\boldsymbol{X}_t$ into a unified feature representation $\boldsymbol{F}^{(0)}$. Then the heterogeneous adjacency matrix $\tilde{\boldsymbol{A}}$ is obtained by (3). Let l denotes the current step in the loop (the depth of the search) and $\boldsymbol{F}^{(l)}$ denotes the feature representation of all nodes at this step. Then, each step in the loop of Algorithm 1 proceeds as follows: first, the hyperparameter β_l can determined by $\log(1/l+1)$. Meanwhile, all nodes aggregate the feature representations of their neighboring nodes into a matrix $\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}$. Then, the aggregated neighboring feature matrix $\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}$ is concatenated with the node's current representation $\boldsymbol{F}^{(l)}$ and the initial unified feature matrix $\boldsymbol{F}^{(0)}$. This concatenated feature matrix is fed into a fully connected (FC) layer with a nonlinear activation function $\sigma(\cdot)$ and model coefficients $(1-\beta_l)\boldsymbol{I}+\beta_l\boldsymbol{W}^{(l)}$, which makes the model more robust. The outputs are used at the next step of the algorithm. Finally, the proposed deep embedding generation algorithm takes the embedding feature matrix $\boldsymbol{F}^{(D)}$ as an output.

Note that only transistors are prone to have aging degradations in analog IC netlists while features of all devices need to be fed into our proposed deep H-GCN to perform the embedding generation algorithm. Thus, a mask matrix with the size *#transistors* $\times$ *#devices* is used to extract the embedding features of each transistor in the last embedding generation layer. Finally, several traditional FC layers are adopted to estimate `dvtlin` of each transistor by inputting the embedding features. The mean square error (MSE) function is used as the loss function to compute the errors between the ground-truth and our estimated `dvtlin`. All model coefficients, including w_r, $\boldsymbol{U}_t$, and $\boldsymbol{W}^{(l)}$ in Algorithm 1, are updated by the back-propagation and the gradient-based method in an end-to-end fashion in each training epoch.

B. Over-Smoothing Analysis

Here, we provide an over-smoothing analysis for the proposed H-GCN in Section IV and the extended deep H-GCN. For simplicity, we assume connection-type-related model coefficients w_r for $\forall r \in \mathcal{R}_{\mathcal{E}_{\text{hmg}}}$ and the unified feature matrix $\boldsymbol{F}^{(0)}$ to be nonnegative, that is, $w_r \geq 0$ and $\boldsymbol{F}^{(0)} \geq \boldsymbol{O}$. Note that we can convert w_r and $\boldsymbol{F}^{(0)}$ to be nonnegative by a linear transformation if they are negative [32].

1) Oversmoothing Analysis of Multilayer H-GCN: To illustrate that the over-smoothing issue in the proposed H-GCN, we will show that the directed multigraph as shown in Section III is strongly connected and aperiodic, and the embedding generation as shown in (4) is equivalently expressed as the formulation of Laplacian smoothing.

To show strong connectivity and aperiodicity for the directed multigraph, we first see the bipartite graph representation of analog IC as shown in Fig. 3. The bipartite graph is connectivity since there is a path between every pair of nodes [42]. Then, it is easily proved in contrapositive form that the directed multigraph is connectivity if it is transferred from a connected bipartite. Moreover, the directed multigraph is strongly connected since any two devices are paired up as a connection in each net to transfer the connected bipartite as the directed multigraph. Besides, the directed multigraph is aperiodic since there is no integer $\kappa > 1$ that divides the length of every cycle of the graph [43].

Here, we will analyze whether the embedding generation as shown in (4) satisfies the formulation of Laplacian smoothing. Since $\boldsymbol{F}^{(0)}$ is nonnegative, like [32], we remove the nonlinear activation function (we use ReLU as the nonlinear activation function). Thus, (4) can be transferred as follows:

$$\boldsymbol{F}^{(l)} = \text{CONCAT}\Big(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\Big) \cdot \boldsymbol{W}^{(l)}. \tag{7}$$

We set $\boldsymbol{W}^{(l)} = [\lambda \boldsymbol{I}, (1-\lambda)\boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)}$. Then, (7) can be transferred as follows:

$$\begin{aligned}\boldsymbol{F}^{(l)} &= \text{CONCAT}\Big(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}\Big) \cdot [\lambda \boldsymbol{I}, (1-\lambda)\boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)} \\ &= \Big((\boldsymbol{I}-\lambda \boldsymbol{I}) \cdot \boldsymbol{F}^{(l-1)} + \lambda \tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}\Big) \cdot \hat{\boldsymbol{W}}^{(l)} \\ &= \Big(\boldsymbol{I} - \lambda \boldsymbol{D}^{-1}\boldsymbol{L}\Big)\boldsymbol{F}^{(l-1)} \cdot \hat{\boldsymbol{W}}^{(l)}\end{aligned} \tag{8}$$

where $0 < \lambda \leq 1$ is a parameter that controls the weighting between the features of the current node and its neighbors. $\boldsymbol{L} = \boldsymbol{D} - \tilde{\boldsymbol{A}}$ is the graph Laplacian matrix, where $\boldsymbol{D}$ is a diagonal matrix with $D(i,i) = \sum_j \tilde{A}(i,j)$. According to (8), the embedding generation $\boldsymbol{F}^{(l)}$ as shown in (4) is equivalently expressed as the formulation of Laplacian smoothing, i.e., $(\boldsymbol{I} - \lambda \boldsymbol{D}^{-1}\boldsymbol{L})\boldsymbol{F}^{(l-1)} \cdot \hat{\boldsymbol{W}}^{(l)}$ [26].

The proposed directed multigraph is strongly connected and aperiodic. Besides, the embedding generation can be equivalently expressed as the formulation of Laplacian smoothing. Thus, according to the fundamental Theorem of Markov Chains [44], the random walk converges to a unique stationary distribution $\boldsymbol{\pi}$, i.e., $\lim_{D\to\infty} \boldsymbol{f}_i^{(D)} = \boldsymbol{\pi}$, where the ith row in $\boldsymbol{F}^{(D)}$ is the ith node's embedding feature $\boldsymbol{f}_i^{(D)}$. In other words, as the number of layers increases, the representations of the nodes in H-GCN are inclined to converge to a certain value and become indistinguishable, which is the oversmoothing issue.

2) Oversmoothing Analysis of Multilayer Deep H-GCN: Here, we give an oversmoothing analysis for the extended deep H-GCN to illustrate it can preventing the oversmoothing even if the number of layers goes to infinity.

We consider the embedding generation with initial residual connections and identity mappings as shown in (5), and (6) in the extended deep H-GCN. Since $\boldsymbol{F}^{(0)}$ is non-negative, we remove the nonlinear operation. Moreover, we fix $(1-\beta_l)\boldsymbol{I} + \beta_l \boldsymbol{W}^{(l)}$ to be $\gamma_l \boldsymbol{W}^{(l)}$, where γ_l is a parameter. Thus, (5), and (6) can be transferred as follows:

$$\boldsymbol{F}^{(l)} = \text{CONCAT}\Big(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(0)}\Big)\gamma_l \boldsymbol{W}^{(l)}. \tag{9}$$

In the same manner, we set $\boldsymbol{W}^{(l)} = [\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}]^\top \cdot \hat{\boldsymbol{W}}^{(l)}$. Then, (9) can be transferred as follows:

$$\begin{aligned}\boldsymbol{F}^{(l)} &= \text{CONCAT}\Big(\tilde{\boldsymbol{A}} \cdot \boldsymbol{F}^{(l-1)}, \boldsymbol{F}^{(l-1)}, \boldsymbol{f}^{(0)}\Big) \cdot [\boldsymbol{I}, \boldsymbol{I}, \boldsymbol{I}]^\top \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)} \\ &= \Big(\Big(\tilde{\boldsymbol{A}} + \boldsymbol{I}\Big)\boldsymbol{F}^{(l-1)} + \boldsymbol{F}^{(0)}\Big) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)}.\end{aligned} \tag{10}$$

Furthermore, we set $\boldsymbol{F}^{(0)} = (\tilde{\boldsymbol{A}} + \boldsymbol{I}) \cdot \hat{\boldsymbol{x}}$. Thus, (10) can be further transferred as follows:

$$\begin{aligned}\boldsymbol{F}^{(l)} &= \Big(\Big(\tilde{\boldsymbol{A}} + \boldsymbol{I}\Big)\Big(\boldsymbol{f}^{(l-1)} + \hat{\boldsymbol{x}}\Big)\Big) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)} \\ &= \Big((\boldsymbol{I} - \boldsymbol{L})\Big(\boldsymbol{f}^{(l-1)} + \hat{\boldsymbol{x}}\Big)\Big) \cdot \gamma_l \hat{\boldsymbol{W}}^{(l)}.\end{aligned} \tag{11}$$

For simplicity, like [32], we further set $\hat{\boldsymbol{W}}^{(l)}$ as an identity matrix. Consequently, according to (11) and [32], we can express the final embedding feature representation as follows:

$$\boldsymbol{F}^{(D)} = \left(\sum_{l=0}^{D}\left(\prod_{k=D-l}^{D}\gamma_k\right)(\boldsymbol{I}-\boldsymbol{L})^l\right)\cdot\hat{\boldsymbol{x}}. \tag{12}$$

This expression suggests that deep H-GCN converges to a distribution that carries information from both the input feature $\hat{\boldsymbol{x}}$ ($\boldsymbol{F}^{(0)}$) and the graph structure $\boldsymbol{L}$ ($\tilde{\boldsymbol{A}}$). This property ensures that deep H-GCN will not suffer from an oversmoothing issue even if the number of layers goes to infinity.

VI. Large Scale Graph Training via Neighborhood Sampling

With the fast development of semiconductors, more and more devices and connections are integrated into an analog IC, which brings great challenges to the model training because of the iterative embedding generations among neighboring nodes. For example, to compute the gradients of node v_i in the lth layer, the embedded features of the neighboring nodes $\mathcal{N}(v_i)$ of v_i are required, i.e., some features in the $(l-1)$th layer. Moreover, the features of $\mathcal{N}(\mathcal{N}(v_i))$ are also needed, i.e., some features in the $(l-2)$th layer. As the search depth D (the number of embedding generation layers) increases, the kind of neighborhood explosion will bring the much more node number in each training epoch so that it causes lots of training time and memory resources even through out-of-memory on GPU. Fig. 7(a) is taken as an example to show the neighborhood explosion.

To achieve enough scalability, a straightforward method is using the traditional graph partitioning algorithms [45] to partition a large-scale graph to be several isolated small-size subgraphs to be trained. However, the features of the neighboring node (device) cannot be aggregated to the node itself in all training epochs if the neighboring node and the node itself are partitioned into two isolated subgraphs. As a result, these graph partitioning algorithms cause performance degradation.

To alleviate the performance degradation, like the dropout technique in CNNs [46], a probability-based sampling method is leveraged to obtain different subgraphs among different training epochs. Considering that the runtime is dominated by graph transformation in our proposed flow, we perform a probability-based sampling method on the bipartite graph, instead of heterogeneous multigraph [25], to achieve significant speedup and good scalability. In our proposed probability-based sampling method, the sampling probability relies on the number of edges (degree) in each node, as shown in the following:

$$\mathcal{P}(v_i) = \frac{D_b(i,i)}{|\boldsymbol{D}_b|} \tag{13}$$

where $\boldsymbol{D}_b$ is a diagonal matrix. Its each diagonal element $D_b(i,i) = \sum_j A_b(i,j)$, where $\boldsymbol{A}_b$ is the adjacency matrix of the bipartite graph (constructed by directly parsing and flattening the circuit netlist as in Fig. 3). $|\boldsymbol{D}_b|$ is summation of all of the elements in the matrix $\boldsymbol{D}_b$. According to (13), the node with more edges is sampled with a higher probability. Note that although the analog IC bipartite graph contains device nodes and net nodes, our proposed probability-based neighborhood sampling algorithm is performed to only sample device nodes while the net nodes will be automatically removed by our proposed heterogeneous multigraph representation as shown in Section III.

Algorithm 2 Neighborhood Sampling Algorithm

Require: Search depth D, the input transistor node set $\mathcal{V}_s$, the number of device nodes to be trained T, analog IC netlist bipartite graph $\mathcal{BG}$;

Initialize device node set $\mathcal{V}^{(0)} \leftarrow \mathcal{V}_s$;
for $k = 1$ to D **do**
  $\mathcal{U}^{(k)} \leftarrow \emptyset$;
  for all $v \in \mathcal{V}^{(k-1)}$ **do**
    $\mathcal{U}^{(k)} \leftarrow \mathcal{U}^{(k)} \cup \mathcal{N}(v)$; ▷ expand the net node set.
  end for
  $\mathcal{V}^{(k)} \leftarrow \emptyset$;
  for all $\mu \in \mathcal{U}^{(k)}$ **do**
    $\mathcal{V}^{(k)} \leftarrow \mathcal{V}^{(k)} \cup \mathcal{N}(\mu)$; ▷ expand the device node set.
  end for
end for
$\mathcal{U}^{(D)} \leftarrow \cup_{k=1}^{D}\mathcal{U}^{(k)}$; ▷ combine the net node sets with $1, 3, \cdots, 2D-1$ hops distance away from one node of $\mathcal{V}_s$.
$\mathcal{V}^{(D)} \leftarrow \cup_{k=0}^{D}\mathcal{V}^{(k)}$; ▷ combine the device node sets with $0, 2, \cdots, 2D$ hops distance away from one node of $\mathcal{V}_s$.
if $|\mathcal{V}^{(D)}| > T$ **then**
  for all $v_i \in \mathcal{V}^{(D)} \backslash \mathcal{V}_s$ **do**
    Calculate $\mathcal{P}(v_i)$ according to Equation 13;
  end for
  Sample $T - |\mathcal{V}_s|$ nodes according to the probability $\mathcal{P}$ to the set $\mathcal{V}_b$;
  $\mathcal{V}_b \leftarrow \mathcal{V}_b \cup \mathcal{V}_s$;
else
  $\mathcal{V}_b \leftarrow \mathcal{V}^{(D)}$;
end if
return The bipartite subgraph with $\mathcal{V}_b \cup \mathcal{U}^{(D)}$ of $\mathcal{BG}$.

The basic idea behind our proposed probability-based sampling method is that the device with more connections has a more important influence on the aging degradation of its neighboring transistors. Unlike traditional graph partitioning algorithms [45], our proposed probability-based algorithm can obtain different subgraphs among different training epochs. In other words, even though the neighboring node and the node itself are partitioned into two isolated subgraphs in the current training epoch, they may be partitioned into the same subgraph in other training epochs. Thus, our proposed probability-based neighborhood sampling algorithm can achieve more robust performance and good scalability.

Our proposed probability-based neighborhood sampling algorithm is summarized in Algorithm 2. We provide the search depth D, the transistor node set $\mathcal{V}_s$, the number of device nodes to be trained T, and analog IC netlist bipartite graph $\mathcal{BG}$ as inputs. First, we need to find a device node set

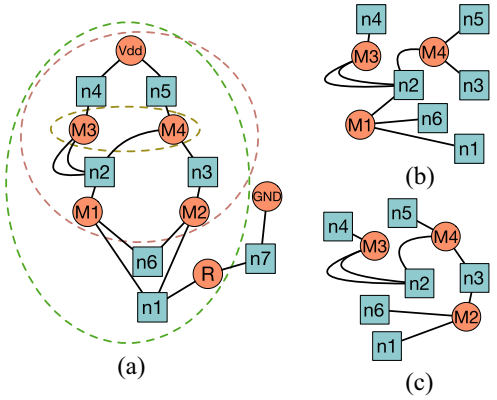

Fig. 7. Neighborhood sampling: (a) Neighborhood expansions. (b) Bipartite subgraph with the device node set $\mathcal{V}_b = \{\texttt{M1}, \texttt{M3}, \texttt{M4}\}$ and the net node set $\mathcal{U}^{(3)} = \{\texttt{n1}, \texttt{n2}, \texttt{n3}, \texttt{n4}, \texttt{n5}, \texttt{n6}\}$. (c) Bipartite subgraph with the device node set $\mathcal{V}_b = \{\texttt{M2}, \texttt{M3}, \texttt{M4}\}$ and the net node set $\mathcal{U}^{(3)} = \{\texttt{n1}, \texttt{n2}, \texttt{n3}, \texttt{n4}, \texttt{n5}, \texttt{n6}\}$.

$\mathcal{V}^{(D)}$ and a net node set $\mathcal{U}^{(D)}$, whose each element is equal or less $2 - D$ hops away from one of the transistor nodes in $\mathcal{V}_s$ (from line 2 to line 13). As mentioned above, the deep H-GCN model brings much more elements in the node set $\mathcal{V}^{(D)}$. In order to sample $T - |\mathcal{V}_s|$ elements from the node set $\mathcal{V}^{(D)}$, the sampling probability of each node is calculated by (13) (from line 15 to line 17). Then, all sampled nodes are combined with the transistor node set $\mathcal{V}_s$. Finally, the algorithm outputs the bipartite subgraph with the device node set $\mathcal{V}_b$ with T device nodes and the net node set $\mathcal{U}^{(D)}$. By performing Algorithm 2 in each training epoch, different bipartite subgraphs can be trained so that our deep H-GCN model can achieve more robust performance. Besides, like the dropout technique in CNNs [46], our proposed probability-based neighborhood sampling algorithm can alleviate overfitting since it randomly drops embedding features from the neighboring nodes during the training stage.

We take Fig. 7 as an example to illustrate the proposed probability-based neighborhood sampling algorithm. In this example, we set the search depth as $D = 3$ and the number of device nodes as $T = 3$. The initial input node set is $\mathcal{V}_s = \{\texttt{M3}, \texttt{M4}\}$. As shown in Fig. 7(a), the 6-hop device node set is $\mathcal{V}^{(3)} = \{\texttt{M1}, \texttt{M2}, \texttt{M3}, \texttt{M4}, \texttt{Vdd}, \texttt{R}\}$ and the 6-hop net node set is $\mathcal{U}^{(3)} = \{\texttt{n1}, \texttt{n2}, \texttt{n3}, \texttt{n4}, \texttt{n5}, \texttt{n6}\}$. Except for M3 and M4 in $\mathcal{V}_s$, we only need to sample one more node from $\mathcal{V}^{(3)} \backslash \mathcal{V}_s$. According to (13), M1 and M2 have the largest probability to be sampled since they have the most edge number. If M1 is sampled, Algorithm 2 takes the subgraph with the device node $\mathcal{V}_b = \{\texttt{M1}, \texttt{M3}, \texttt{M4}\}$ and net node $\mathcal{U}^{(3)}$ shown in Fig. 7(b) as an output. If M2 is sampled, Algorithm 2 takes the subgraph with the device node $\mathcal{V}_b = \{\texttt{M2}, \texttt{M3}, \texttt{M4}\}$ and net node $\mathcal{U}^{(3)}$ shown in Fig. 7(c) as an output. The two subgraphs are trained in different training epochs.

Note that here we perform neighborhood sampling method on the bipartite graph, instead of heterogeneous graph used in our previous work [25]. In this way, we can achieve significant speedup and facilitate training on a large-scale graph since the graph transformation is the runtime bottleneck. Besides, our proposed neighborhood sampling algorithm can be naturally used in the inference stage.

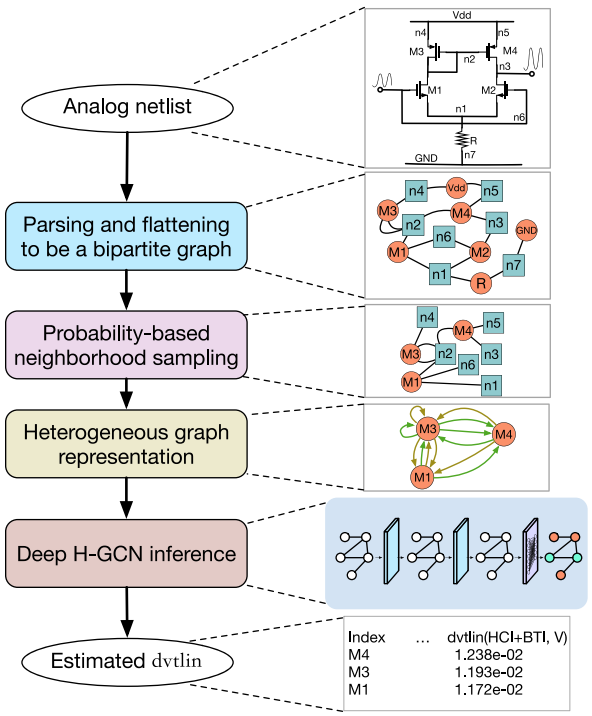

Fig. 8. Overall flow.

VII. Overall Flow

The overall flow of our proposed fast analog IC aging-induced degradation estimation framework is illustrated in Fig. 8, consisting of four parts: 1) parsing and flattening netlist; 2) probability-based neighborhood sampling; 3) heterogeneous graph representation; and 4) deep H-GCN inference. Our flow takes the netlist with static stress conditions as an input. Then, the netlist is parsed and flattened to be a bipartite graph. In order to achieve enough scalability, the probability-based neighborhood sampling, as discussed in Section VI, is adopted to obtain a subgraph. Moreover, the subgraph is transferred to be the heterogeneous graph to efficiently represent the topology of analog ICs, as explained in Section III. The heterogeneous graph is input into the deep H-GCN to perform dvtlin estimation. Finally, the overall flow takes the estimated dvtlin values of the target transistors as outputs. Based on the techniques mentioned above, the proposed flow is expected to achieve fast and accurate dvtlin estimation.

VIII. Experimental Results

A. Benchmarks and Experimental Setting

Considering that the aging-related wear-out is very severe at the 5-nm technology node in the industry, our experiments are illustrated on eight different industrial phase-locked loop designs implemented at the very advanced 5-nm technology node. It should be noted that our proposed method is very general. It does not rely on any technology information or parameters so that it can be adopted in other technologies. These designs are represented with post-layout netlists. The post-layout netlist files consist of Spectre and SPICE formats [47], as inputs in our models. We run an industrial aging

TABLE I
STATISTICS OF DESIGNS

Design	#trans.	#device	#net
1	4,348	99,009	18,155
2	4,382	99,696	18,299
3	3,999	179,758	31,303
4	3,998	185,480	33,819
5	4,980	692,480	111,308
6	523	31,279	6,002
7	6,398	452,109	76,807
8	1,998	96,749	16,006

TABLE II
DEVICE TYPE

Type	#Param.
MOS	51
MOS spice	75
DIO/ESD	8
Cap	12
R	6
VSource	1

DFR tool to obtain the `dvtlin` value of each transistor. Note that in each design, the stress conditions are given by very sophisticated designers to obtain `dvtlin` as a golden-truth value. To evaluate the estimation performance, we use seven designs for training and the remaining designs for testing each time.

The statistics of the eight industrial phase-locked loop designs are shown in Table I, where the netlist (Design 5) has up to 692K devices. The netlist of each design is parsed and flattened. Then all alternating current (ac) voltage sources (dynamic conditions), such as behavioral signal (Verilog-A description), sine wave, pulse and piecewise linear (pwl), are ignored while all dc voltage sources are considered so that our model can be independent of the dynamic stress conditions. Except for ac voltage sources, there are 32 types of basic devices in all designs. According to their parameters, all these devices are divided into six categories, as listed in Table II. Their feature vectors have different lengths, i.e., the #Param. Correspondingly, six matrices are adopted to map feature vectors into a unified latent space.

According to the domain knowledge, four pins of transistors and two pins of diodes play an important role in circuit analysis [39]. Consequently, in the heterogeneous graph representation, all edges connecting to these six types of pins are emphasized by categorizing them into six types of edges. All edges connecting to other pins are uniformly treated as another type of edges. In total, seven adjacency matrices are used to specify various connection pins, i.e., gate, drain, source, substrate, anode, cathode, and others.

Two traditional graph learning models, our proposed H-GCN and the extended deep H-GCN, are implemented in the experiments, as listed in Table III. GCN [16] and GCNII [32] models use the concatenated features representation and treat the analog IC netlist as homogeneous undirected multigraphs mentioned in Section III. However, our H-GCN [25] and deep H-GCN use the proposed heterogeneous directed multigraph

TABLE III
GRAPH LEARNING MODEL

Method	Feature		Graph representation	
	Concat.	Latent	Homo.	Heter.
GCN	✓		✓	
GCNII	✓		✓	
H-GCN-concat	✓			✓
H-GCN		✓		✓
Deep H-GCN-concat	✓			✓
Deep H-GCN		✓		✓

representation and the unified latent space mapping algorithm. In our H-GCN and deep H-GCN models, the ReLU function is used as the activation function [40]. We use MSE between the ground truth and our estimated `dvtlin` as the loss function with weight decay hyperparameter 10^{-7} and stochastic gradient descent for optimization. We set the embedding generation with search depth $D = 2, 4, 6, 8$ and three FC layers in the four graph learning models. Moreover, we set the device sampling size $T = 1000$ in Algorithm 2. In order to illustrate the performance of graph learning models, all inferences are repeatedly performed ten times to report averages. Each output feature size of embedding generation layers and the latent space mapping layer is 512. The output feature sizes of three FC layers are set as 4096, 1024, and 1, respectively. The batch size is 32, and the number of the training epoch is 600. The same settings and configurations are used in GCN [16] and GCNII [32]. To improve numerical stability, we normalize all feature vectors and adjacency matrices. In order to illustrate the performance of these data-driven graph learning models, we also run dynamic and static aging reliability simulations using the industrial aging DFR tool.

The proposed graph representation is implemented with Python and Networkx library [48]. All graph learning models are implemented with TensorFlow [49] and are trained and tested on a Linux machine with 18 cores and NVIDIA Tesla V100 GPU with 32-GB memory.

Mean absolute error (MAE) defined in (14) is used as a metric to evaluate the absolute accuracy on the testing set

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \hat{y}_i\right| \tag{14}$$

where i indicates the ith transistor. y_i is the ith transistor's `dvtlin`, as the golden-truth value, obtained by the industrial aging DFR tool with the appropriate stress given by very sophisticated designers. $\hat{y}_i$ is the estimated `dvtlin` of the ith transistor by graph learning methods or the static reliability simulation. n is the number of transistors on the testing set. In addition, we use the r^2 score [50] defined in (15) as a metric to evaluate the relative accuracy on the testing set

$$r^2\text{score} = 1 - \frac{\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{15}$$

where $\bar{y}$ denotes the mean of the golden-truth values y_i on the testing set. r^2 score indicates how the regression prediction

TABLE IV
MAE (MV) AND r^2 SCORE COMPARISONS

Design	DFR tool (static)		GCN [16]		GCNII [32]		H-GCN [25]		Deep H-GCN	
	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score	MAE	r^2 Score
1	4.009	0.181	1.316	0.703	1.332	0.691	0.914	0.821	**0.824**	**0.843**
2	4.072	0.194	1.389	0.596	1.339	0.619	0.893	0.814	**0.856**	**0.839**
3	4.543	0.327	4.070	0.588	4.166	0.599	2.302	0.817	**2.012**	**0.840**
4	4.515	0.332	4.111	0.588	4.177	0.551	2.575	0.746	**2.350**	**0.815**
5	4.160	0.277	3.750	0.521	4.021	0.354	2.525	0.787	**2.454**	**0.816**
6	3.962	0.395	2.077	0.802	2.092	0.802	1.661	0.834	**1.541**	**0.865**
7	4.319	0.266	3.166	0.685	3.168	0.689	2.889	0.826	**2.704**	**0.874**
8	4.594	0.224	3.491	0.637	3.748	0.610	2.670	0.786	**2.503**	**0.840**

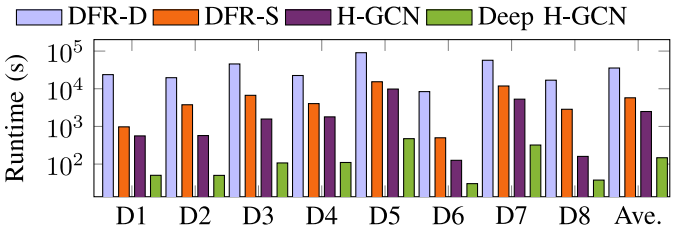

Fig. 9. Runtime comparisons.

perfectly fits the data. The higher r^2 and lower MAE mean the better accuracy.

B. Accuracy and Runtime

We train the four graph learning models in Table III using embedding generation with search depth $D = 2, 4, 6, 8$ and list their best accuracies in Table IV. Besides, we also list the accuracies of the static aging reliability simulation by using the industrial aging DFR tool. Compared with graph learning models, the static aging reliability simulation causes the worst accuracy since it completely ignores dynamic stress conditions. Thus, our proposed data-driven approach with a supervised learning manner is effective to estimate aging-induced degradation.

According to Table IV, the typical GCN [16] and GCNII [32] have the worse accuracy since they do not consider the heterogeneity of the analog circuit. Compared with our proposed H-GCN, our extended deep H-GCN can achieve a more accurate estimation since it can extract information from multihop devices without an oversmoothing issue. The same reason explains why GCNII [32] outperforms the typical GCN [16].

We compare the runtime of our extended deep H-GCN with H-GCN [25], static, and dynamic aging simulations by using the industrial DFR tool as shown in Fig. 9, where DFR-D and DFR-S denote the dynamic and static aging reliability simulations, respectively. The static and dynamic aging reliability simulations need to perform fresh simulation, stress simulation, aging simulation as illustrated in Fig. 2. Thus, they consume the longest time. Moreover, compared with static aging reliability simulations, the dynamic aging reliability simulation consumes a longer runtime since it needs a large number of accepted transient steps. Compared with the traditional dynamic and static aging reliability simulations, our proposed methods do not need computationally expensive fresh simulation, stress simulation or aging simulation. Thus, our H-GCN and extended deep H-GCN can achieve significant speedup. According to Fig. 9, our extended deep H-GCN can achieve 241× and 39× speedup on average, respectively, comparing with the dynamic and static aging reliability simulations.

Compared with our H-GCN, our extended deep H-GCN consumes less runtime even if it has more embedding generation layers (search depth). Since our extended deep H-GCN performs the neighborhood sampling on the bipartite graph to alleviate graph transformation overhead. According to Fig. 9, our extended deep H-GCN can achieve 16× speedup on average. In particular, `dvtlin` values of all transistors in an industrial post-layout netlist with 692K devices (Design 5) are estimated within 472 s. Thus, our proposed deep H-GCN has good scalability to handle the large-scale analog IC post-layout netlists.

C. Ablation Study

In this section, we give two ablation studies to illustrate the effectiveness of the deep structures and our proposed the unified latent space mapping algorithm.

1) Effectiveness of the Deep Structures and Oversmoothing: The accuracies w.r.t the search depth of embedding generation $D = 2, 4, 6, 8$ among different graph learning models are shown in Fig. 10. As the search depth of embedding generation increases, both GCN [16] and our H-GCN [25] bring performance degradation because of the oversmoothing issue. In other words, they achieve the best accuracy with a shallow structure. While as the search depth of embedding generation increases, GCNII [32] and our deep H-GCN can achieve a more accurate estimation. Thus, GCNII [32] and our extended deep H-GCN can alleviate the oversmoothing issue. However, compared with GCNII [32], our extended deep H-GCN can characterize the heterogeneity of the analog circuit so that it can achieve a more accurate estimation.

2) Effectiveness of the Unified Latent Space Mapping Algorithm: We compare the accuracies of H-GCN and deep H-GCN with H-GCN-concat and deep H-GCN-concat, respectively. Here, H-GCN-concat and deep H-GCN-concat adopt the concatenated feature representation as listed in Table III. For comparisons, according to Fig. 10, we choose the best

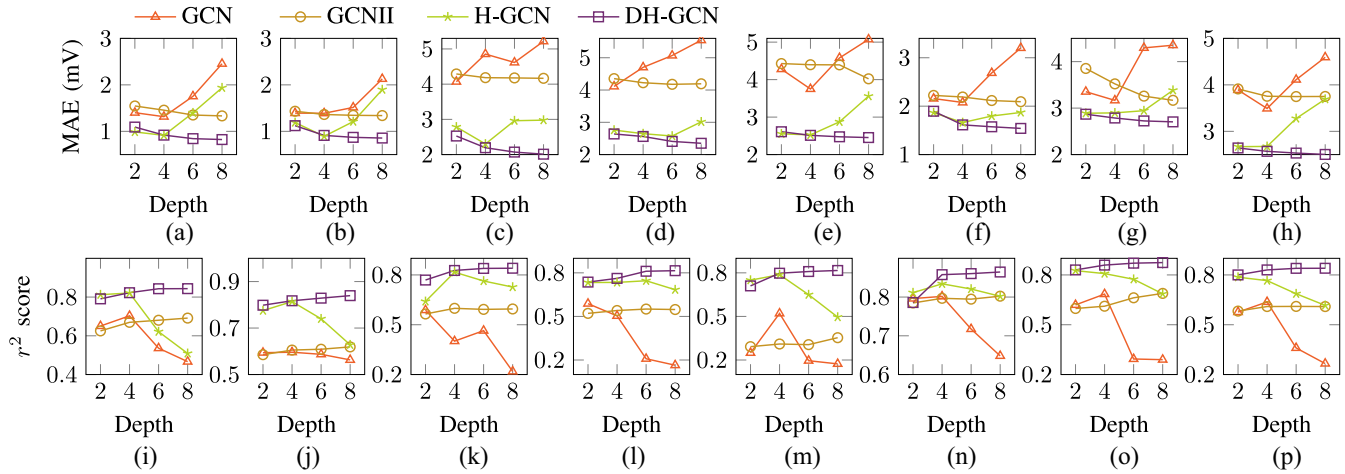

Fig. 10. Embedding generation depth versus accuracy: (a)–(h): MAE on design 1 to design 8. (i)–(p): r^2 Score on design 1 to design 8 (DH-GCN denotes our extended deep H-GCN).

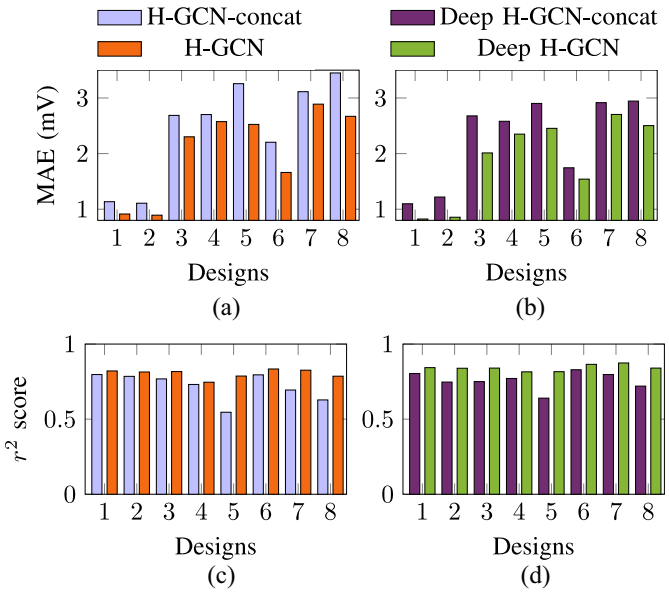

Fig. 11. MAE and r^2 Score between HGCN and HGCN-concat and between deep H-GCN and deep H-GCN-concat: (a) MAE between H-GCN and H-GCN-concat. (b) MAE between deep H-GCN and deep H-GCN-concat. (c) r^2 Score between H-GCN and H-GCN-concat. (d) r^2 Score between deep H-GCN and deep H-GCN-concat.

configurations (the search depth of embedding generation D) for each model on each design. MAE and r^2 score among HGCN, HGCN-concat, deep H-GCN and deep H-GCN-concat are shown in Fig. 11. Our proposed unified latent space mapping algorithm can improve accuracy on both HGCN and deep H-GCN since it can effectively exploit and encode features.

IX. Conclusion

In this article, we proposed an H-GCN to fast estimate aging-induced transistor degradation in analog ICs. To characterize the multityped devices and connection pins, a heterogeneous directed multigraph was adopted to efficiently represent the topology of analog ICs. A latent space mapping method was used to transform the feature vector of all typed devices into a unified latent space. We further extended the proposed H-GCN to be a deep version via initial residual connections and identity mappings. The extended deep H-GCN can extract information from multihop devices without an oversmoothing issue. A probability-based neighborhood sampling method on the bipartite graph was adopted to ease the model training on large-scale graphs and achieve good scalability. We conducted experiments on very advanced 5-nm industrial benchmarks. Compared with traditional graph learning methods and the static aging reliability simulations by using an industrial DFR tool, our proposed deep H-GCN can achieve more accurate estimations of aging-induced transistor degradation. Compared with the dynamic and static aging reliability simulations, our extended deep H-GCN can achieve on average 241× and 39× speedup, respectively. Thus, our proposed deep H-GCN can significantly improve the efficiency of aging verification.

Acknowledgment

The authors would like to thank Dr. Chao Yang from HiSilicon for helpful discussions.

References

[1] D. Sengupta and S. S. Sapatnekar, "Estimating circuit aging due to BTI and HCI using ring-oscillator-based sensors," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 10, pp. 1688–1701, Oct. 2017.

[2] Z. Yu, Z. Sun, R. Wang, J. Zhang, and R. Huang, "Hot carrier degradation-induced dynamic variability in FinFETs: Experiments and modeling," *IEEE Trans. Electron Devices*, vol. 67, no. 4, pp. 1517–1522, Apr. 2020.

[3] J. B. Bernstein, M. Gurfinkel, X. Li, J. Walters, Y. Shapira, and M. Talmor, "Electronic circuit reliability modeling," *Microelectron. Rel.*, vol. 46, no. 12, pp. 1957–1979, 2006.

[4] T. Chen, B. Lin, H. Geng, S. Hu, and B. Yu, "Leveraging spatial correlation for sensor drift calibration in smart building," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 7, pp. 1273–1286, Jul. 2021.

[5] T. Chen, B. Lin, H. Geng, and B. Yu, "Sensor drift calibration via spatial correlation model in smart building," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6.

[6] K. B. Sutaria *et al.*, "Duty cycle shift under static/dynamic aging in 28nm HK-MG technology," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Monterey, CA, USA, 2015, pp. 1–5.

[7] R. H. Tu *et al.*, "Berkeley reliability tools-BERT," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 12, no. 10, pp. 1524–1534, Oct. 1993.

[8] M. B. Yelten, P. D. Franzon, and M. B. Steer, "Surrogate-model-based analysis of analog circuits—Part II: Reliability analysis," *IEEE Trans. Device Mater. Rel.*, vol. 11, no. 3, pp. 466–473, Sep. 2011.

[9] C. Schlünder *et al.*, "From device aging physics to automated circuit reliability sign off," in *Proc. IEEE Int. Rel. Phys. Symp. (IRPS)*, Monterey, CA, USA, 2019, pp. 1–12.

[10] Q. Sun, A. A. Rao, X. Yao, B. Yu, and S. Hu, "Counteracting adversarial attacks in autonomous driving," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1–7.

[11] T. Chen *et al.*, "An efficient sharing grouped convolution via Bayesian learning," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Jun. 10, 2021, doi: 10.1109/TNNLS.2021.3084900.

[12] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1175–1187, Jun. 2019.

[13] H. Geng *et al.*, "Hotspot detection via attention-based deep layout metric learning," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1–8.

[14] R. Chen *et al.*, "Faster region-based hotspot detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Sep. 3, 2021, doi: 10.1109/TCAD.2020.3021663.

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

[16] W. L. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2017, pp. 1024–1034.

[17] Y. Ma, Z. He, W. Li, L. Zhang, and B. Yu, "Understanding graphs in EDA: From shallow to deep learning," in *Proc. ACM Int. Symp. Phys. Design (ISPD)*, 2020, pp. 119–126.

[18] Y. Ma *et al.*, "High performance graph convolutionai networks with applications in testability analysis," in *Proc. ACM/IEEE Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6.

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

[20] X. Gao, C. Deng, M. Liu, Z. Zhang, D. Z. Pan, and Y. Lin, "Layout symmetry annotation for analog circuits with graph neural networks," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, 2021, pp. 152–157.

[21] K. Kunal *et al.*, "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Proc. IEEE/ACM Design Autom. Test Eur. (DATE)*, Grenoble, France, 2020, pp. 55–60.

[22] Y. Li *et al.*, "A customized graph neural network model for guiding analog IC placement," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, 2020, pp. 1–9.

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

[24] S. Chang, W. Han, J. Tang, G.-J. Qi, C. C. Aggarwal, and T. S. Huang, "Heterogeneous network embedding via deep architectures," in *Proc. ACM Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2015, pp. 119–128.

[25] T. Chen, Q. Sun, C. Zhan, C. Liu, H. Yu, and B. Yu, "Analog IC aging-induced degradation estimation via heterogeneous graph convolutional networks," in *Proc. IEEE/ACM Asia South Pac. Design Autom. Conf. (ASPDAC)*, Tokyo, Japan, 2021, pp. 898–903.

[26] Q. Li, Z. Han, and X.-M. Wu, "Deeper insights into graph convolutional networks for semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, vol. 32, 2018, pp. 3538–3545.

[27] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 5453–5462.

[28] Y. Rong, W. Huang, T. Xu, and J. Huang, "Dropedge: Towards deep graph convolutional networks on node classification," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2020, pp. 1–17.

[29] F. Wu, A. H. D. Souza, Jr., T. Zhang, C. Fifty, T. Yu, and K. Q. Weinberger, "Simplifying graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2019, pp. 6861–6871.

[30] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2018, pp. 1–15.

[31] J. Klicpera, S. Weißenberger, and S. Günnemann, "Diffusion improves graph learning," in *Proc. Conf. Neural Inf. Process. Syst. (NIPS)*, 2019, pp. 13354–13366.

[32] M. Chen, Z. Wei, Z. Huang, B. Ding, and Y. Li, "Simple and deep graph convolutional networks," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 1725–1735.

[33] (2016). *New Generation Reliability Model*. [Online]. Available: http://www.mos-ak.org/berkeley_2016/publications/T11_Xie_MOS-AK_Berkeley_2016.pdf

[34] (2018). *Re-Thinking Reliability Analysis*. [Online]. Available: https://indico.esa.int/event/222/contributions/2089/attachments/1779/2077/AMICSA_06_18_18_ReThinking_Reliability_Analysis.pdf

[35] M. Meissner and L. Hedrich, "FEATS: Framework for explorative analog topology synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 2, pp. 213–226, Feb. 2015.

[36] L. Wu *et al.*, "Scalable global alignment graph kernel using random features: From node embedding to graph embedding," in *Proc. ACM Int. Conf. Knowl. Discov. Data Min. (KDD)*, 2019, pp. 1418–1428.

[37] M. Zwerger, M. Neuner, and H. Graeb, "Analog power-down synthesis," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 36, no. 12, pp. 1954–1967, Dec. 2017.

[38] M. Zwerger, M. Neuner, and H. Graeb, "Power-down circuit synthesis for analog/mixed-signal," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 656–663.

[39] B. Razavi, *Design of Analog CMOS Integrated Circuits*. New Delhi, India: Tata McGraw-Hill Educ., 2002.

[40] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 30, 2013, p. 3.

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

[42] R. Diestel, *Graph Theory*. New York, NY, USA: Springer-Verlag, 2005.

[43] J. Jarvis and D. R. Shier, "Graph-theoretic analysis of finite Markov chains," in *Applied Mathematical Modeling: A Multidisciplinary Approach*. Boca Raton, FL, USA: Chapman Hall, 1999, p. 85.

[44] D. Randall, "Rapidly mixing Markov chains with applications in computer science and physics," *Comput. Sci. Eng.*, vol. 8, no. 2, pp. 30–41, Mar./Apr. 2006.

[45] C.-E. Bichot and P. Siarry, *Graph Partitioning*. Hoboken, NJ, USA: Wiley, 2013.

[46] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[47] K. Kundert, *The Designer's Guide to Spice and Spectre®*. New York, NY, USA: Springer, 2006.

[48] *Networkx: Network Analysis in Python*. Accessed: Nov. 2019. [Online]. Available: https://networkx.github.io

[49] *TensorFlow: An End-to-End Open Source Machine Learning Platform*. Accessed: Nov. 2019. [Online]. Available: https://www.tensorflow.org

[50] N. R. Draper and H. Smith, *Applied Regression Analysis*, vol. 326. New York, NY, USA: Wiley, 1998.

Tinghuan Chen (Graduate Student Member, IEEE) received the B.Eng. and M.Eng. degrees in electronics engineering from Southeast University, Nanjing, China, in 2017 and 2014, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His research interests include machine learning in analog/mixed signal VLSI design-for-reliability and cyber–physical systems.

Qi Sun (Graduate Student Member, IEEE) received the B.Eng. degree in computer science from Xidian University, Xi'an, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

His current research interests include deep neural network hardware acceleration, high-level synthesis, and design space exploration.

Canhui Zhan received the M.Eng. degree in material physics from the South China University of Technology, Guangzhou, China, in 2004.

Since 2005, he has been engaged in VLSI design-for-reliability, design-for-testability, and failure mechanism research and developed the DFR tool with HiSilicon Technologies Company, Shenzhen, China.

Changze Liu (Member, IEEE) received the Ph.D. degree in electronics engineering from Peking University, Beijing, China, in 2013.

Then, he joined the Semiconductor Division, Samsung Electronics, where he involved in the development of 14/10/7-nm processes. In 2017, he joined Huawei HiSilicon, Shenzhen, China, as technical experts for product R&D in advanced technology node. In 2020, he served as the CEO of Huawei Technologies Research and Development Belgium NV, Leuven, Belgium. He has authored and coauthored over 30 scientific papers, including IEEE IEDM, VLSI, and IRPS.

Dr. Liu served for the Technical Program Committee of IEEE IEDM, EDTM, IRPS, ISCAS, IPFA, and ESREF.

Huatao Yu, photograph and biography not available at the time of publication.

Bei Yu (Member, IEEE) received the Ph.D. degree from The University of Texas at Austin, Austin, TX, USA, in 2014.

He is currently an Associate Professor with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong.

Dr. Yu received seven best paper awards from ASPDAC 2021, ICTAI 2019, Integration, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS in 2018, ISPD 2017, SPIE Advanced Lithography Conference 2016, ICCAD 2013, ASPDAC 2012, and six ICCAD/ISPD Contest Awards. He has served as the TPC Chair of ACM/IEEE Workshop on Machine Learning for CAD, and in many journal editorial boards and conference committees. He is the Editor of IEEE TCCPS Newsletter.