Machine Learning in Advanced IC Design: A Methodological Survey

Tinghuan Chen
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong

Bei Yu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong

Grace Li Zhang
Chair of Electronic Design Automation
Technical University of Munich
80333 Munich, Germany

Bing Li and Ulf Schlichtmann
Chair of Electronic Design Automation
Technical University of Munich
80333 Munich, Germany

Editor's notes:
The increasing complexity and size of design space poses significant challenges for integrated circuit (IC) design. This article discusses the potential of machine learning (ML) methods to address these challenges and provides a comprehensive survey of the current state of knowledge along both IC design problems and ML-based solutions. The article also summarizes the open problems at the intersection of advanced IC design and ML.
—Janardhan Rao (Jana) Doppa, Washington State University

With An Aggressive scaling of the CMOS technology, the number of transistors integrated into a design exponentially increases. The IC design flow contains architectural design, circuit design, physical design, verification, manufacture, packaging, security, and power management, as shown in Figure 1. The large-scale integration and complicated design flow lead to enormous challenges in IC designs. For example, costly explorations in huge design spaces are required at the circuit design stage to obtain implementations that satisfy all design specifications and achieve optimal performance. Besides, the time-consuming simulation at the verification stage, such as lithography simulation, leads to low efficiency for large-scale IC designs.

To address the challenges described above, machine-learning (ML) techniques have been employed in IC designs and have already achieved impressive success in various applications. Due to its learning ability from data, cutting-edge research takes advantage of MLML to improve the performance and efficiency of traditional optimization algorithms. The advantage of ML in IC designs is that it provides an accurate and efficient performance evaluation in several IC design stages. Besides, ML transforms the traditional analytical simulation and optimization problem into a data-to-data mapping problem, which opens a door for IC developers with limited knowledge backgrounds.

This article provides a comprehensive survey of ML methodologies in advanced IC design. Unlike existing survey articles [1], [2], [3], [4], which

Digital Object Identifier 10.1109/MDAT.2022.3216799
Date of publication: 25 October 2022; date of current version: 20 January 2023.

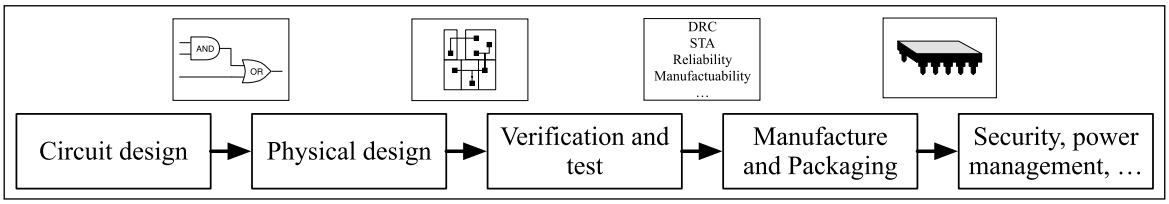

Figure 1. IC design flow.

introduce relevant research by sequential design steps, this article summarizes the state-of-the-art research based on the taxonomy of ML methodologies. The benefits of this survey are threefold: 1) it shows a development course where feature representations of IC designs are learned more efficiently and accurately, along with the development of ML methodologies; 2) the development of ML methodologies in advanced IC designs is promoted by the state-of-the-art ideas of customization for vanilla learning methods from the methodology view; and 3) the potential insights are inspired to address more IC design challenges by borrowing ML methodologies from resolved problems.

Overview

ML techniques learn a model from sample data (training data) to make decisions or predictions on unseen data [5]. Traditional ML algorithms mainly focus on learning a prediction model. The features must be manually extracted in such an algorithm. Then these features are input into the model to generate prediction results [5]. Such ML models are shallow since they rely on handcrafted features, instead of learning features from the model itself.

Deep models have been introduced to automatically extract features from raw data by the model itself [6]. Such models adopt several nonlinear feature conversions from the original raw data. Most deep models are based on deep neural networks (DNNs), where neurons are connected by synapses into a network, and neurons are organized into layers. DNNs can extract and abstract features layer by layer to outperform shallow models in dealing with difficult tasks, for example, IC design.

There are many types of DNNs, for example, convolutional neural networks (CNNs) and graph neural networks (GNNs). CNNs are widely used to extract effective features from image-based data [7], while GNNs are used to obtain an effective feature representation from irregular grid-based data [8]. Based on various DNNs, deep generative models [9], [10] and deep reinforcement learning techniques [11] are developed. With approximating a statistical distribution with DNNs, deep generative models are used to generate new samples satisfying a specific probability distribution [9], [10]. Deep reinforcement learning incorporates various DNNs to help agents learn how to reach their goals [11]. They are also widely used and customized to guide IC designs effectively and have already achieved a good performance.

In the following sections, we will systematically and comprehensively introduce ML methodologies and their applications in IC designs. These ML methodologies include shallow models, CNNs, GNNs, generative models, and reinforcement learning. ML methodologies and their cutting-edge applications in IC designs are categorized as shown in Table 1.

Shallow models

Shallow models, as shown in Figure 2, are traditional ML algorithms. The handcrafted features from waveforms, circuits, or layout characteristics are preprocessed to convert as a feature vector $\boldsymbol{x}_i$. Typical preprocessing methods include normalization and dimension reduction and increase. Then this shallow model f is then used to predict the label $\hat{y}_i$.

To determine the parameters $\boldsymbol{w}$ of the shallow model, this model should be trained with a training data set. The training objective of the shallow model is to minimize the average difference between model predictions and real labels of all samples [5], which can be formulated as follows:

$$\min_{\boldsymbol{w}} \mathbb{E}_{(\boldsymbol{x}, y) \sim \mathcal{P}_r(\boldsymbol{x}, y)}[\mathcal{L}(y, f(\boldsymbol{x};\boldsymbol{w}))] \quad (1)$$

where y is the ground-truth label, $P_r(\boldsymbol{x},y)$ is the real data distribution, and $\mathcal{L}$ is the loss function to evaluate the difference between model predictions and real labels. According to f, the shallow models can be classified as linear and nonlinear.

Table 1. ML methodologies and IC designs.

ML methodology		Stage in IC design flow
Shallow model	Linear	Verification [12], [13], Security [14], [15], Power management [16], [17]
	Nonlinear	Circuit design [18]–[24], Physical design [25], Verification [26], [27], Power management [28]
CNNs	Conv.	Physical design [29], [30], Verification [31]–[34]
	Attention	Verification [35], [36]
GNNs	Graph conv.	Physical design [37], [38], Verification [39], Test [40]
	Graph attention	Physical design [41], [42]
	Graph pooling	Circuit design [43], Physical design [44], Verification [45]
Generative model	VAE	Physical design [46]
	GAN	Physical design [47], [48], Verification [49], Manufacture [50], [51], Packaging [52]
RL	Value function	Power management [53]–[55]
	Policy function	Circuit design [56], Physical design [57]
	Actor-critic	Circuit design [58]–[60], Physical design [61]

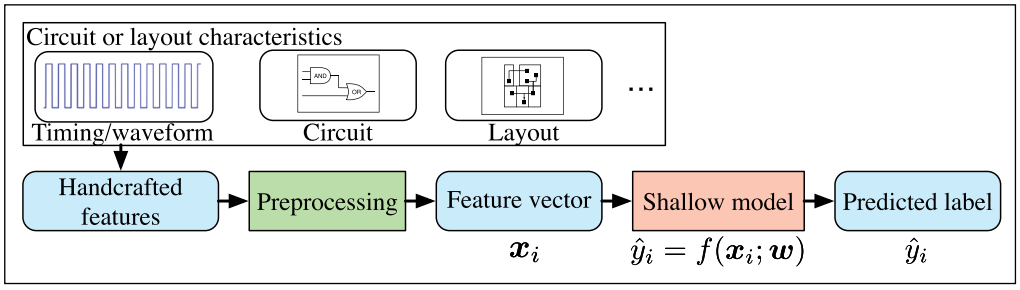

Figure 2. Shallow learning-based framework.

Linear models

1) *Background:* Typical linear models include linear regression and logistic regression. Figure 3a illustrates a linear model, where the input feature x_i is first multiplied with the model parameter w_i and the multiplication result is accumulated to predict the corresponding label. The linear model can be easily fit or trained with the least-square method.
2) *Application to IC designs:* Hardware attacks are a hardware security and reliability concern. Traditional mitigation schemes bring a high implementation overhead. A linear model is used to fast detect and identify hardware attacks in [14] and [15]. The memory access data and memory behaviors are selected as the handcrafted features. Due to the low computational complexity and simplicity, the linear model is deployed on the chip for real-time detection and identification with low implementation overhead.

Power management has an extra implementation overhead for a multicore system. To reduce implementation overhead, Clark et al. [16] proposed an on-chip power management design, where a linear model is adopted to predict future traffic load for routers. Several network throughput parameters are selected as the handcrafted features. However, the typical linear model assumes that parameters have independent effects on the predicted value.

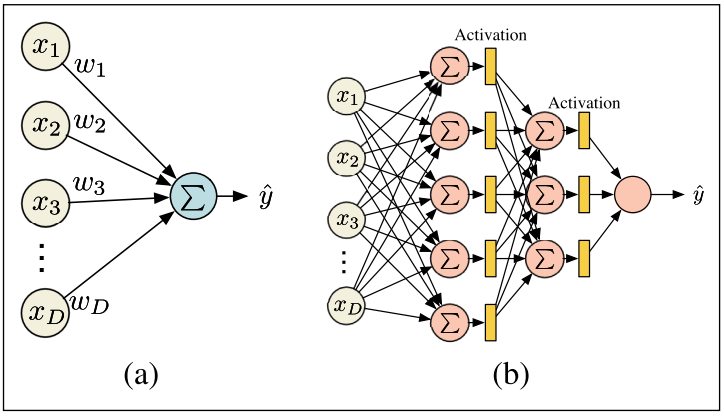

Figure 3. (a) Linear model. (b) ANN.

The variations in a linear regression model have a different impact on the predicted power. A binning approach is developed to predict task performance on all core types [17]. A workload throughput and power prediction model contains different linear layers, where the output of each layer relies on the memory or compute metrics.

Due to its simplicity, the linear model is used at the verification stages, such as timing and reliability. Considering strong correlations among different processes, voltage, and temperature (PVT) analysis corners, and a linear regression model is proposed to predict the timing performance of unobserved PVT corners [12]. The path information, cell types, and clock specifications are selected as handcrafted features to input into the model. Then the model outputs an estimated delay value in the path. Logistic regression is employed to detect electromigration (EM) violations [13]. A logistic sigmoid acts on a typical linear regression for binary classification. The netlist-level net-specific and layout-level neighborhood-related information is selected as handcrafted features to input into the model.

Nonlinear models

1) *Background:* The relationship between extracted features and predictions is complicated. Linear models fail to provide such a complicated capability that nonlinear models can address. The typical nonlinear models include artificial neural networks (ANNs) [5] as shown in Figure 3b, Gaussian process (GP) [62], nonlinear support vector machines (SVMs) [63] and random forest [64]. ANNs use a nonlinear activation function, such as rectified linear unit (ReLU) and Sigmoid, as shown in Figure 4a, to enhance the nonlinear representation. Nonlinear SVM and GP use a kernel function, as shown in Figure 4b, to implicitly map samples from the original feature space to a high-dimensional space, where the linear inseparability problem in the original feature space can be solved. However, proposing a suitable kernel function for a specific task is difficult. An alternative scheme is the adaptive basis function model. Random forest is an adaptive basis function model, where the basis functions are the regions of input features and the weights are given in each region [64].
2) *Application to IC designs:* Nonlinear models have been widely applied in IC designs to capture complicated and nonlinear correlations. The increasing design-verification iterations are caused by complicated design rules and constraints at the advanced process node. A nonlinear SVM is proposed to predict detailed-route design rule checking (DRC) violations after global routing so that DRC violations can be predicted without a runtime-intensive detailed route [26]. A layout is partitioned into several small grids. Then several netlist and layout parameters, such as connected pins and cell/pin density, are selected as handcrafted features. The SVM model finally outputs the DRC hotspot prediction for each grid. An ANN model is used to rapidly evaluate layout feasibility by estimating the correlation among all the sensitive interconnect parasitics [27]. There is a high nonlinear relationship between hardware configuration and performance. Analytical-based performance evaluations are computationally expensive. Zhuo et al. [19] and Cao et al. [28] employed a support vector regression (SVR) for fast power estimation by taking input signal and hardware configuration as inputs. In [18], the nonlinear SVR and GP regressions are employed to estimate adder performances. The regressors replace the industry electronic design automa-

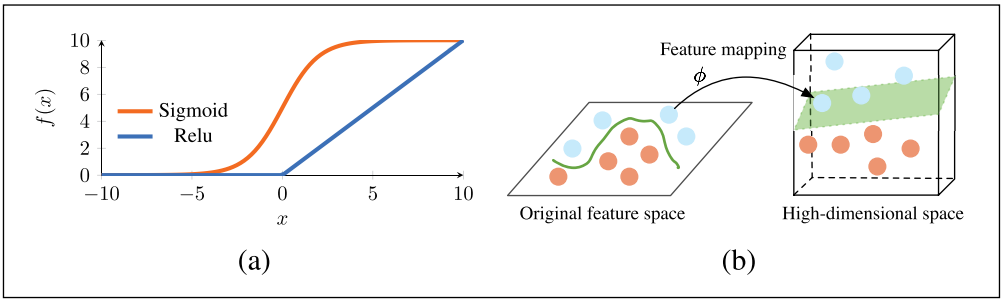

Figure 4. (a) Activations. (a) Feature space mapping by the kernel.

tion (EDA) tool to fast obtain power, delay, and area. The circuit structure information and tool settings are selected as handcrafted features.

Unlike other shallow nonlinear models, GP can provide predictions with uncertainties. The prediction and uncertainty can be combined as a metric to efficiently collect training data, which is Bayesian optimization. In other words, the training data set is expanded by selecting the most informative data at each iteration. To reduce the runtime of the EDA tool and improve learning efficiency at the design stage, Bayesian optimization-based design space exploration techniques are proposed to fast and efficiently determine design parameters and EDA tool settings for digital design [20], [21], [22], analog design [23], and hardware deployment [24].

Regression forest has been employed in IC designs. A regression forest models the relationship between the Pareto hypervolume of search trajectory and design to make a better tradeoff among design objectives [25]. Pareto hypervolume is used as a metric to evaluate the quality of a solution set for multiobjective optimization [20].

Convolutional Neural Networks

A layout naturally represents an image consisting of design information. Deep-learning models are introduced to extract features from a layout by the mode itself. As shown in Figure 5, CNNs are one of such models, which consists of convolutional layers, pooling layers, and fully-connected (FC) layers [6]. This section introduces the two basic types of layers and two advanced modules.

Typical CNN structure

1) *Background:* The motivation of a convolutional layer is to extract features from layouts, as shown in Figure 5. Figure 6a shows the convolution operations, where the size of the input feature is $M \times N \times D$. The small tensor with the size $U \times V \times D$ is called a filter. A slice of a filter is called a kernel. Multiple filters modify the input feature by slicing through it left to right and top to bottom with multiply-accumulate operations. The result of such operations is adjusted by an activation function to generate the output feature map with size $M' \times N' \times P$.

To reduce the required amount of computations in the subsequent layers, a pooling layer is usually added after a convolutional layer. Figure 6b illustrates two types of pooling, maxpooling, and meanpooling. Both types of pooling operations first partition the feature map into several blocks. After that, maxpooling (meanpooling) takes the maximum (average) value in a block as the value in the modified feature map.

2) *Application to IC designs:* In recent years, CNNs have been widely used in IC designs. Traditional analytical-based manufacturability and reliability verifications are performed on the layout, which is time-consuming. In [31], CNNs detect lithography hotspots fast, as shown in Figure 5. The layout is partitioned into several small blocks (tiles). Afterward, each block's discrete cosine transform coefficients are encoded as a feature tensor and fed into a CNN model to detect lithography hotspots. However, only small blocks are input into this hotspot detector to detect lithography hotspots in its central region. To deal with a large-scale layout, lots of computational resources will be consumed. Chen et al. [32] proposed a faster two-stage region-based lithography hotspot detection framework, which can mark multiple hotspot locations within a region whose size is much larger than a block (tile). Unlike the manufacturability issue, the IR-drop reliability issue relies on time-related factors. In [33], power density distribution is input into a CNN to predict the maximum IR drop. A maximum module is designed to lead to the peak IR drop.

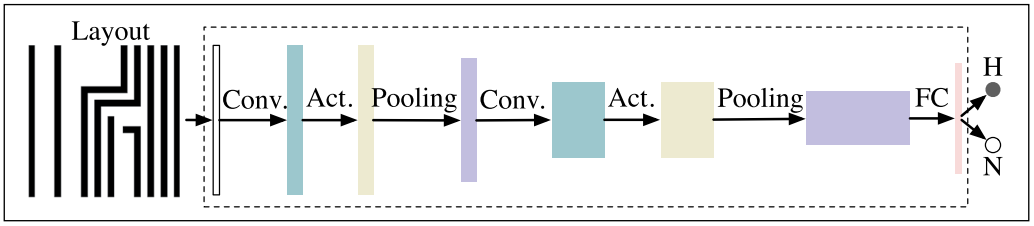

Figure 5. Typical CNN structure.

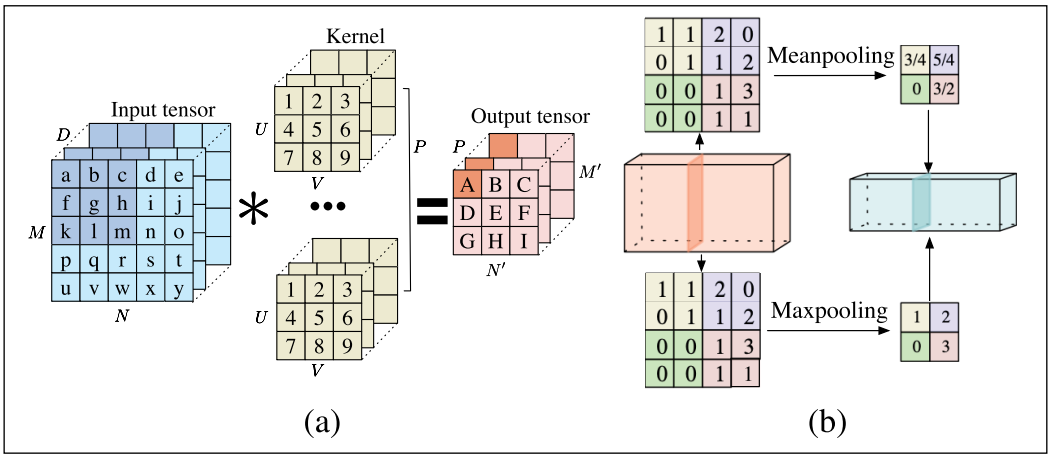

Figure 6. (a) Convolution operation. (b) Pooling operations.

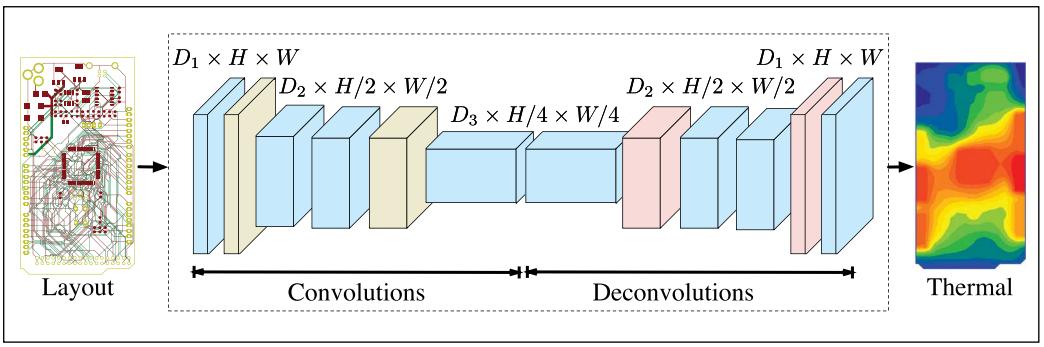

Figure 7. FCN.

The different designs have different layout sizes. There is not enough data set to train a model for each size. A feasible scheme is that a model supports varying size inputs. To support varying size input tensors or predict a violation heatmap, as shown in Figure 7, all FC layers are replaced with convolutional layers and deconvolutional layers to construct a fully convolutional network (FCN) [65]. The deconvolutional layers are implemented via various interpolation methods. In industry, global routing is performed for DRC hotspot prediction after the placement stage. However, trial global routing is time-consuming. In [34], FCN is used to predict DRC hotspots without global routing. Specifically, the 3-D input tensor consists of macros, for example, intellectual property macros, global long-range rectangular uniform wire density (RUDY), and global RUDY pins. A well-trained FCN-based model is expected to predict performance and guide design to facilitate design closure. For the performance prediction model, the gradients with respect to the input layout represent the sensitivity of performance. In [29], a congestion heatmap prediction model is added to the typical cost function to guide routing. In [30], a routability heatmap prediction model is added to the typical cost function to guide placement. The routability-aware placement is formulated as a deep-learning training problem so that it can be handled by deep-learning toolkits on advanced hardware platforms.

Advanced modules

1) *Background:* Attention is one of the advanced modules to enhance feature representation [66]. It assigns different weights to each part of the input and extracts more critical information to enable the model to make more accurate judgments. The attention maps are calculated from channel and space dimensions. In Figure 8a, a shared multilayer perceptron (MLP) is used to extract two features from the outputs of two pooling branches in the channel attention module.

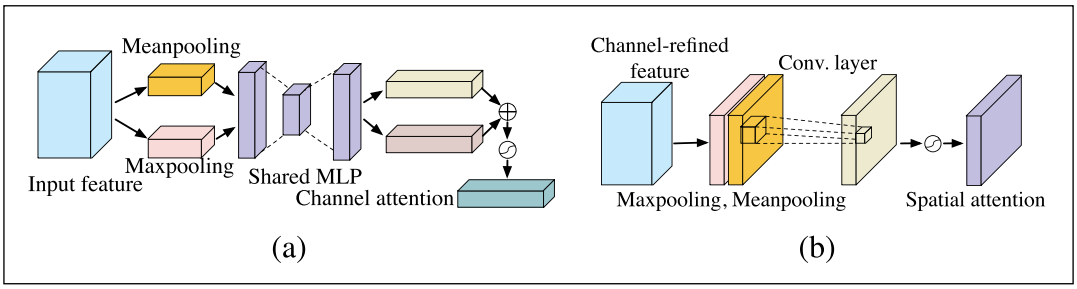

Figure 8. Attention (a) channel attention and (b) spatial attention.

In Figure 8b, the maxpooling, meanpooling, and convolutional layers are stacked to obtain spatial attention from a channel-refined feature. Based on the attention mechanism, the transformer is proposed in many sequence-to-sequence tasks [67]. They both contain multihead attention modules to extract information at different locations globally.

2) *Application to IC designs:* Attention is a lightweight general-purpose module so that it is easy to be integrated into various CNNs. To enhance efficient representations of layout features, the attention module is integrated into CNNs to detect lithography hotspots [35]. The attention module sequentially generates a 1-D channel attention map and a 2-D spatial attention map. The obtained attention maps are multiplied with the input feature via a broadcasting mechanism. To mark multiple hotspot locations within a region in a large-scale layout, a one-stage detector consisting of a transformer module is proposed to identify lithography hotspots [36]. The output is the coordinates and size of the bounding box that lithography hotspots within.

Graph Neural Networks

A netlist represents a graph. For example, as shown in Figure 9, each device is represented as a node, and each interconnection is represented as an edge. The design parameters of each device, such as channel length and width for the transistor, are used as features. An embedding method generates new features by aggregating original features from the neighborhoods to the node itself. The embedding methods followed by typical neural networks form GNNs. The embedding methods contain graph convolution, graph attention (GAT), and graph pooling.

Graph convolution

1) *Background:* A typical graph convolution leverages the spatial relationship among nodes to aggregate information and generates node embedding [8]. For example, two graph convolutional layers are used to generate node embedding of node M1 from all nodes within a 2-hop distance, as shown in Figure 9. In practice, the feature dimension relies on the number of device parameters, such as channel length and width for the transistor. Here, we assume that a feature dimension is a concrete number. In the first layer, the input feature dimension of each node is assumed as 10. For node M2, the aggregation input is a matrix, whose size is 6×10, since there are five neighborhood nodes and the node M2 itself. Then an aggregation operation, such as mean operation over nodes, is performed to reduce node dimension to 10. The aggregated feature vector is the input into an encoding module to extract feature information and obtain a feature vector with 100 dimensions.
2) *Application to IC designs:* In IC designs, many works try to use typical GNNs with graph convolution, named graph convolutional networks (GCNs) and their variants. To avoid signal mismatch, some specific circuit structures and devices must be identified as constraints at the analog physical design stage. The typical method relies on manual identification, which needs rich domain knowledge. To automatically identify these structures and devices, in [37], all devices and their pins are represented as nodes. Then a typical GCN model is used to annotate pairwise constraints. Kunal et al. [38] proposed GANA to create circuit hierarchy trees and classify circuits into subblocks for analog circuits.

GCNs are customized for IC characteristics to improve modeling ability. Traditional analytical

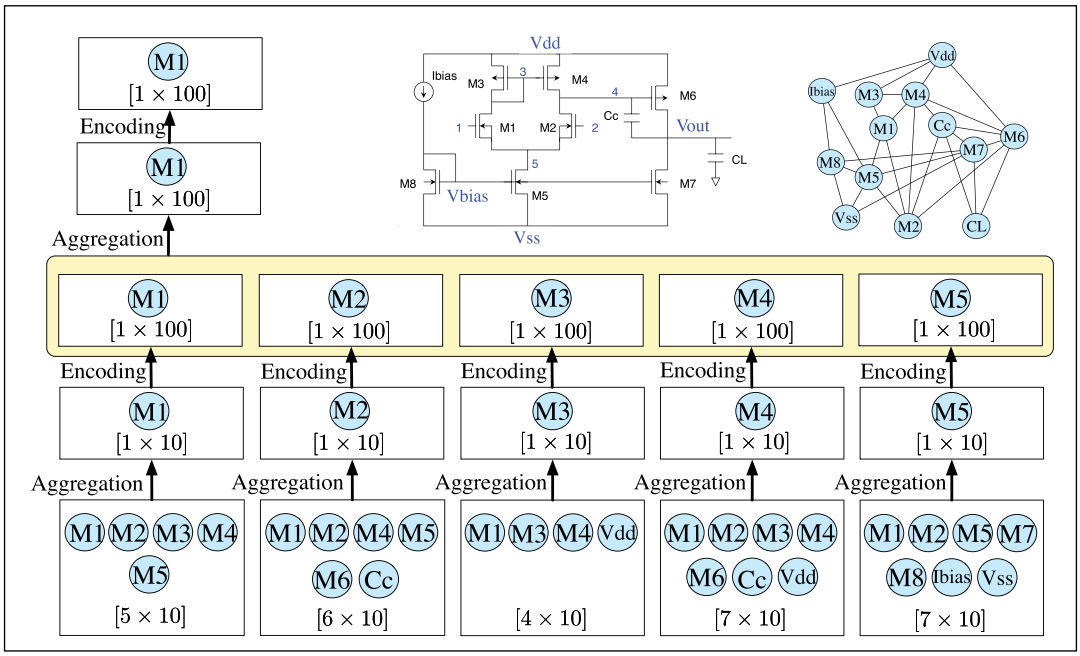

Figure 9. Netlist, graph representation, and graph convolution.

methods at the netlist level are computationally expensive at the verification and test stage. Ma et al. [40] customized a GCN classifier to fast insert observation points into netlists for testing. The digital netlist is transferred into a directed graph, where the edge direction is defined as the signal propagation direction. Then two learnable model coefficients are assigned to distinguish different signal propagation directions in the embedding layer. Based on this idea, a heterogeneous GCN is developed fast to identify aging-prone transistors at the analog netlist level since the analog netlist is heterogeneous [39]. A heterogeneous directed multigraph is used to represent an analog netlist by assigning different model coefficients.

Graph attention

1) *Background:* The typical graph convolution may fail to learn an efficient node embedding from a graph with dense connections. To discriminately aggregate the feature information, the learnable weights, named attention maps, are assigned to each edge to indicate the importance, which is the GAT mechanism [68]. As shown in Figure 10a, an attention map $\alpha_{i,j}$ is obtained by an FC layer, where feature vectors of two interconnected nodes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ with parameter matrix $\boldsymbol{W}$ are used as inputs and $a_1,a_2,\ldots,a_{2m}$ are neurons. The softmax enhances the nonlinear representation. To enhance the modeling ability, GAT introduces multiple attention maps $\alpha_{i,j}$ to aggregate feature information along edges, as shown in Figure 10b.
2) *Application to IC designs:* GAT embedding technology is customized to improve inductive learning ability in IC designs. Inductive learning is reasoning from observed designs to general rules. A model needs an excellent inductive learning ability to apply different and new designs. Device parameters, net parasitics, and length significantly have a significant influence on IC performance. In traditional design flow, they are obtained with physical design. A ParaGraph with GAT is proposed to predict net parasitics and device parameters at the netlist level without physical design [41]. Previous node embeddings are concatenated with the aggregated neighbor embeddings. Different edge types are independently grouped. A self-attention layer is added between the aggregations of each group to improve inductive learning ability. In [42], GNNs with GAT are customized to estimate preplacement net length by capturing a more efficient feature representation of the node and its neighbors. GNNs scale the neighbors' contribution and GAT uses learnable weights to decide the contribution of nodes. Both source and target nodes' features are concate-

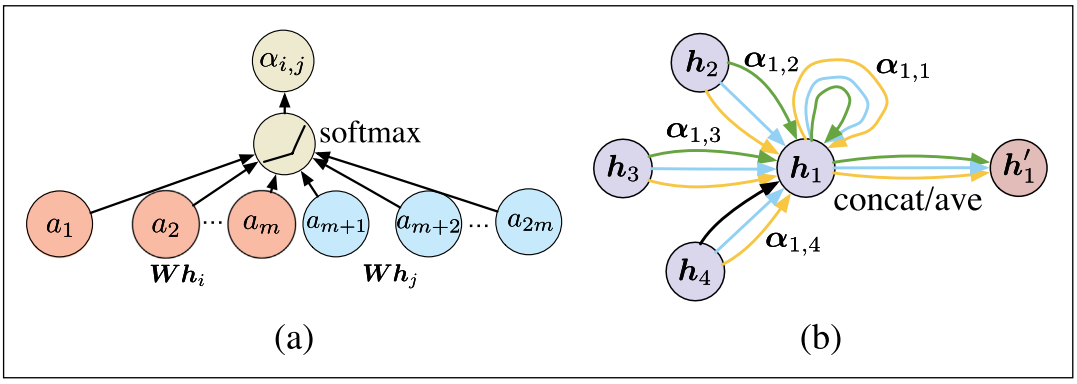

Figure 10. (a) Attention mechanism. (b) Multihead attention.

nated with their edge features to input into edge convolution. The embedding results of all layers are input into the last layer to capture global and local information.

Graph pooling

1) *Background:* Unlike node embedding, graph pooling encodes the whole graph as a feature vector. An effective method is DIFFPOOL, which clusters nodes and abstracts each cluster as a node [69]. Multiple DIFFPOOL layers gradually reduce the node number until there is only one node.
2) *Application to IC designs:* The SPICE-based simulation is costly to obtain circuit performances, such as gain for the analog circuit. It is desired to use the graph learning method to fast and accurately predict performances without SPICE simulation. The DIFFPOOL-based graph learning models are employed to predict analog circuit performances [43], [44]. Then the models guide placement [44] and sizing [43] to facilitate design closure. In monolithic 3-D IC, the surface roughness will produce voids in the dielectric, resulting in delay defects. Delay-fault diagnosis provides early feedback to the foundry and facilitates yield learning. The DIFFPOOL-based graph learning framework utilizes the circuit netlist and failure log files to fast diagnose delay-fault [45].

Generative Models

In some ML tasks, distribution needs to be learned from the training data to generate new samples whose distribution is the same as the training data. These tasks can be performed by the generative model, which contains variational autoencoders (VAEs) [9] and generative adversarial networks (GANs) [10].

Variational autoencoder

1) *Background:* The latent variable $\boldsymbol{z}$ is introduced to model dependencies among different dimensions of $\boldsymbol{x}$. Thus, a model $\mathcal{P}(\boldsymbol{x}\,|\,\boldsymbol{z})$ needs to transfer the distribution of $\boldsymbol{z}$ to $\boldsymbol{x}$. To learn the real distribution $\mathcal{P}_r(\boldsymbol{x})$ from training data, instead of one-to-one mapping, a model $\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x})$ is used to transfer the distribution of $\boldsymbol{x}$ to $\boldsymbol{z}$. To extract the most information from the training data $\boldsymbol{x}$, the model $\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x})$ is predetermined to be independent and identically distributed, whose means and covariances are modeled by neural networks. A VAE [9] consists of an inference network $f_I(\boldsymbol{x};\boldsymbol{\phi})$ (transfer the distribution of $\boldsymbol{x}$ to $\boldsymbol{z}$) and a generative network $f_G(\boldsymbol{z};\boldsymbol{\theta})$ (transfer the distribution of $\boldsymbol{z}$ to $\boldsymbol{x}$), as shown in Figure 11. The inference network approximates distribution $\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x};|\,\boldsymbol{\phi})$. The generative network approximates distribution $\mathcal{P}(\boldsymbol{x}\,|\,\boldsymbol{z};\boldsymbol{\theta})$. $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ denote the model parameters. $\boldsymbol{z}$ and $\hat{\boldsymbol{x}}$ are obtained via sampling from approximated distributions $\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x};|\,\boldsymbol{\phi})$ and $\mathcal{P}(\boldsymbol{x}\,|\,\boldsymbol{z};\boldsymbol{\theta})$, respectively. By introducing a prior $\mathcal{P}(\boldsymbol{z};\boldsymbol{\theta})$, the objective of VAE is to maximize the likelihood

$$\mathbb{E}_{\boldsymbol{z}\sim\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x};\boldsymbol{\phi})}[\log\mathcal{P}(\boldsymbol{x}\,|\,\boldsymbol{z};\boldsymbol{\theta})] - \mathcal{KL}(\mathcal{Q}(\boldsymbol{z}\,|\,\boldsymbol{x};\boldsymbol{\phi}),\mathcal{P}(\boldsymbol{z};\boldsymbol{\theta})) \quad (2)$$

where $\mathcal{KL}(\cdot,\cdot)$ denotes the Kullback–Leibler divergence, which measures the difference between two distributions. Formulation (2) is optimized via a gradient-based method.

2) *Application to IC designs:* There usually exist many implementations of the same design to achieve

similar performance. For example, given a post-placement layout, it is possible that different routing results may achieve similar performance [46]. Analog physical design has complicated design constraints to improve signal transmission quality and prevent mismatches. Typical design-verification iterations bring a low design efficiency. A typical VAE is used to predict routing regions by mimicking the sophisticated manual routing approaches [46]. The routing-related features, such as pins of the entire design and interested nets, are extracted from placement layouts to input into VAE, as shown in Figure 11. VAE captures the human design experience and knowledge. Moreover, the predicted routing region probability map is added to the objective to guide performance-driven routing.

It is necessary to assume that $\mathcal{Q}(\boldsymbol{z}|\boldsymbol{x})$ is an explicit distribution family, whose parameter distributions are approximated by a neural network. However, this assumption limits the capabilities of the neural network.

Generative adversarial networks

1) *Background:* To use neural networks to construct an implicit distribution instead of an explicit parameterized distribution family, GANs use adversarial training to make samples generated from neural networks satisfy the real distribution [10]. As shown in Figure 12, GANs consist of a discriminator network and a generative network. Here, we assume that GANs are used to generate a legal mask. The former judges whether a sample (mask) is from real data (reference legal mask) or the generative network. The latter generates samples (masks) whose source cannot be distinguished by the discriminant network. The discriminator network $f_D(\boldsymbol{x};\boldsymbol{\phi})$ is to distinguish a sample $\boldsymbol{x}$ from the real distribution $\mathcal{P}_r(\boldsymbol{x})$ or the generative model $f_G(\boldsymbol{z};\boldsymbol{\theta})$. To train the discriminant network, in the minimizing cross-entropy manner, the loss function is maximized with respect to $\boldsymbol{\phi}$ as follows:

$$\begin{aligned}&\mathbb{E}_{\boldsymbol{x}\sim\mathcal{P}_r(\boldsymbol{x})}[\log f_D(\boldsymbol{x};\phi)]\\&+\mathbb{E}_{\boldsymbol{z}\sim\mathcal{P}(z)}[\log(1-f_D(f_G(\boldsymbol{z};\boldsymbol{\theta});\phi))]\end{aligned} \tag{3}$$

where $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are model parameters in the discriminant network and generative network. The loss function of the generative network is defined as

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{z}\sim\mathcal{P}(\boldsymbol{z})}[\log(1-f_D(f_G(\boldsymbol{z};\boldsymbol{\theta});\phi))]. \tag{4}$$

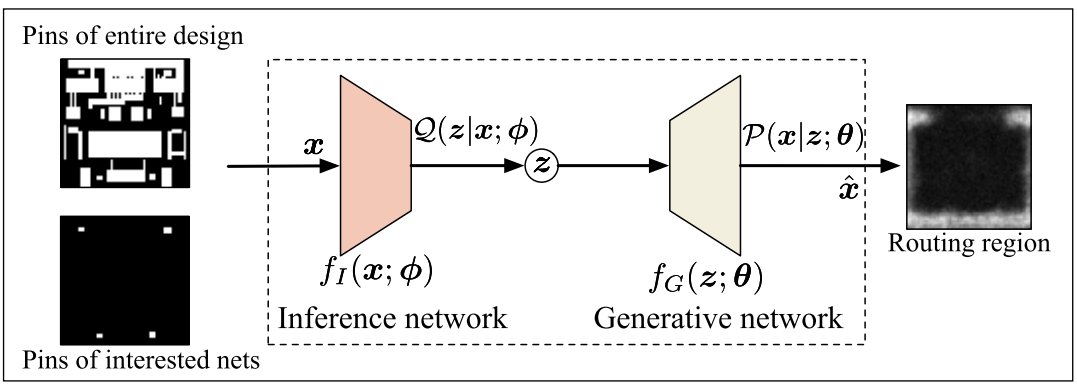

Figure 11. VAE.

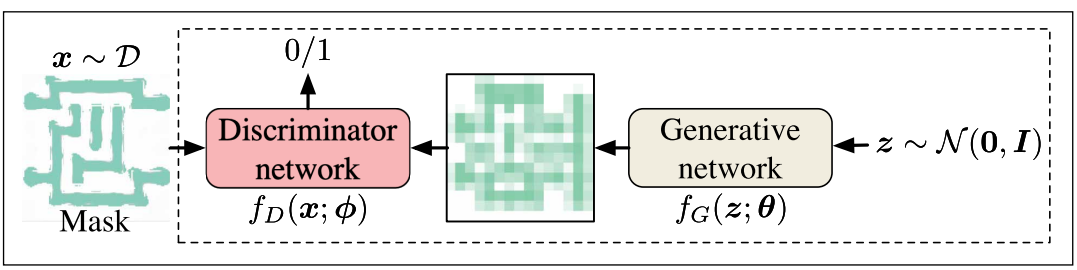

Figure 12. GANs.

2) *Application to IC designs:* The noise sensors need to be placed at suitable locations to monitor voltage emergencies and judge whether the power specification for packaging is satisfied or not. Traditional methods need a massive number of samples to achieve accurate placement. Due to its generative ability, a GAN is developed to efficiently produce more noise maps with a limited number of samples [52].

Unlike the aforementioned typical GAN, the conditional GAN (CGAN) is proposed to learn how to generate fake samples with a specific condition, instead of random noise $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0},\boldsymbol{I})$ [70]. Compared with the typical GAN, CGAN requires the generated features not only to fool the discriminator network, but also to be close to the ground truth.

The CGAN is customized for IC designs at the manufacture, verification, and physical design stages. Lithography proximity effects must be compensated by inserting assist features and correcting mask pattern shapes. Traditional lithography simulation is drastically time-consuming. Yang et al. [50] proposed a CGAN model to achieve good mask optimization without expensive lithography simulation. It does not ensure the generator obtains a high-quality mask. The discriminator performs prediction on target-mask pairs instead of masks. To achieve a high-resolution layout mask optimization, a robust high-resolution CGAN model is presented to perform mask optimization [51], where three subdiscriminators are used to perform at three different scales, respectively.

The high-quality clock trees need to be synthesized by optimizing key desired power, wirelength, and so on. However, a huge number of candidate parameters have to be searched. Liu et al. proposed to utilize CGAN to perform the clock tree synthesis (CTS) optimization and classification tasks [47]. The conditional input can allow the model to refine the generative network and optimize unseen designs. The model can recommend parameter sets to designers, which leads to optimized clock trees. EM-induced IR drop is one of the major failure effects for power grid networks. The traditional EM-induced IR drop analysis is very expensive. CGAN is proposed to fast analyze EM-induced IR drop in power grid networks [49]. The time variable is used as the condition for the generator and discriminator. The discriminator network is developed to distinguish the voltage maps output from the generative network or the real EM-induced voltage map. In analog layout, well generation is essential in establishing the bulk regions. However, traditional manual well generation requires a careful design with rich experience to satisfy performances and design specifications. In [48], CGAN is developed to generate the layout result with generated wells automatically. One customization of CGAN is that the input and output share the geometries in the placement result.

Reinforcement Learning

RL enables an agent to learn an optimal policy via "trial and error" in an interactive environment to achieve the maximum reward [11]. RL is formulated as a Markov decision process (MDP) with a 4-tuple $(\mathcal{S},\mathcal{A},\mathcal{P},\mathcal{R})$, where $\mathcal{R}$ is a reward. $\mathcal{A}$ is a set of actions taken by the agent. $\mathcal{S}$ is a set of states. $\mathcal{P}$ is the probability of transition from one state to another by taking an action. A policy $\pi(a \mid s)$ decides the next action a under the current state s. When the state is s_t at a timeslot t, the agent takes an action $a_t \in \mathcal{A}$ to transit itself to a new state s_{t+1}) with a reward r_{t+1}. The state transition probability is defined as $\mathcal{P}(s_{t+1} \mid s_t, a_t, \ldots,$

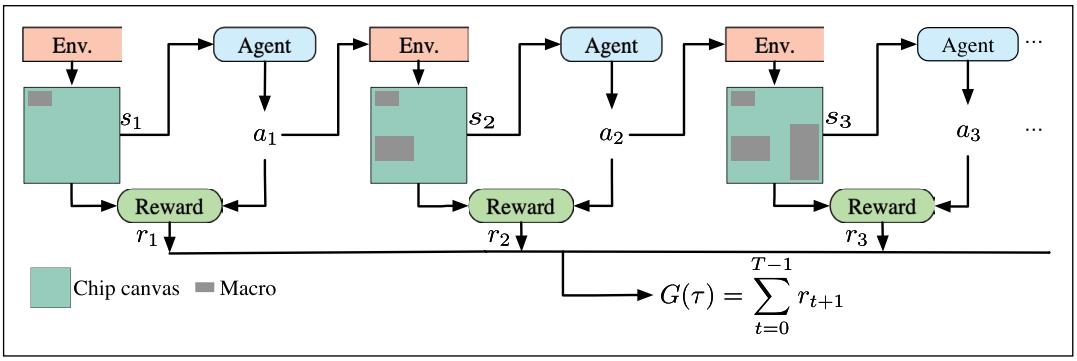

Figure 13. MDP.

$s_0, a_0) = \mathcal{P}(s_{t+1} \mid s_t, a_t)$. As shown in Figure 13, when the initial state is s_0, the agent takes an action a_0 to the state s_1 with a reward r_1. Here, we take the macro placement as an example. Each state is a placement layout and each action is selecting a macro to place a location. Given a policy $\pi(a \mid s)$, MDP is a trajectory $\tau = s_0, a_0, s_1, r_1, a_1, \ldots, r_{T\text{-}1}, a_{T\text{-}1}, s_T, r_T$, whose probability is $\mathcal{P}(s_0)\prod_{t=0}^{T-1} \pi(a_t \mid s_t)\mathcal{P}(s_{t+1} \mid s_t, a_t)$. T is the last timeslot index. The total reward of the trajectory is defined as $G(\tau) = \sum_{t=0}^{T-1} r_{t+1}$, where r_{t+1} is the reward.

According to the RL target, its objective is obtaining a policy $\pi(a \mid s)$ to maximize the expected reward $\mathcal{J}(\boldsymbol{\theta})$

$$\mathcal{J}(\boldsymbol{\theta}) = \mathbb{E}_{\tau \sim \mathcal{P}_{\boldsymbol{\theta}}(\tau)}[G(\tau)] = \mathbb{E}_{s \sim \mathcal{P}(s_0)}\left[V^{\pi}(s)\right] \quad (5)$$

where $\boldsymbol{\theta}$ is learnable parameters in the policy function. V^{π} (s) is the state value function, which denotes the expected total reward by performing the policy π from the state s. s_0 denotes the initial state of the trajectory τ. According to the Markov property, the state-action value function can indicate the total reward when the policy and action are performed at a state [11]. To maximize the expected reward, value function-based methods, policy function-based methods, and actor–critic algorithms are used to determine an optimal policy $\pi(a \mid s)$.

Value function-based methods

1) *Background:* The value function evaluates policy π. Since there is larger cardinality in the state set S and action set A, the exhaustive evaluation of all policies brings a low efficiency. An iteration method is more feasible, where the value function and policy are optimized, alternatively.
2) *Application to IC designs:* Dynamic voltage frequency scaling (DVFS) technique is used to effectively and dynamically save power on chips. In [53], an RL method is used to assign voltage and frequency to each core to improve global performance under a power budget. Each state consists of instructions, current power, voltage, and frequency. The reward is defined as the core throughput indicating the preference for the action that achieves high performance. The policy is chosen to maximize the reward. Moreover, neural networks approximate the value function with continuous states and actions. In [54], a generated network approximates the value for each DVFS algorithm, and a target network is used to train the parameters for producing the target values. Typical value function-based methods need to sufficiently explore the large state space to learn an optimal policy, which causes a long convergence time. In [55], the value function parameters are initially obtained from offline data and incrementally updated on new training examples. As a result, the convergence time is reduced. Meanwhile, the research is performed within a local neighborhood of the state as predicted by the current policy.

However, the value function-based methods may cause a relatively significant change in the value function when the policy is updated. It has a negative influence on convergence.

Policy function-based methods

1) *Background:* Instead of optimizing the value function, policy function-based methods directly search for an optimal policy in the policy space without the help of the value function. Thus, they can overcome the drawbacks of value function-based methods.
2) *Application to IC designs:* Topology synthesis and placement are critical tasks. Topology synthesis is to find an optimal analog circuit structure under specified constraints. Placement is used to place cells and macros on the layout. Traditional analytic-based methods and heuristics may cause suboptimal results and time-consuming optimization processes. An RL-based topology synthesis method is proposed to automatically find an optimal analog circuit structure [56]. All complete or incomplete circuit topology is encoded as a state. Action is defined as selecting a basic block from the predetermined library to connect to the current circuit structure. The reward relies on circuit completeness and performance. An RL-based placement method is developed to automatically place macros [57]. As shown in Figure 13, a partial placement solution is a state, and placing a macro is an action. The reward is defined as the weighted summation of congestion of the placement and proxy wirelength. The policy gradient method is used to update policy network parameters.

However, policy function-based methods are difficult to obtain enough samples when there is a large search space. It is easy to get stuck in the local optimum.

Actor–critic algorithm

1) *Background:* The actor–critic algorithm combines the value function-based and policy function-based methods. The policy function is modeled as an actor to update a policy in the direction given by the critic. The value function is modeled as a critic to evaluate the actor (the current policy function). Both policy and value functions are parameterized with neural networks so that they can be updated at each iteration.
2) *Application to IC designs:* Transistor sizing has a significant influence on circuit performance. Traditional sizing relies on empirical and manual decisions or analytical models. However, empirical and manual decisions may bring suboptimal performance and analytical-based methods have poor generalization. An actor–critic algorithm is proposed to perform sizing in a given circuit schematic [58]. The transistor sizing is formulated as a continuous space search problem. A sequence-to-sequence model encodes the states to find a sizing solution. The action is the predicted size of each transistor, such as channel length and width. The reward function consists of design specifications and hard constraints. An encoder–encoder framework is constructed by an off-policy actor–critic algorithm [71] to map the states to actions. Later on, GNNs are used to enhance agents in [59] since the circuit topology can naturally represent a graph. A shared FC layer with a device-specific encoder is used as the first layer of the critic to encode different actions. The same actor–critic algorithm is developed to perform parameter tuning in logic synthesis [60] and placement [61].

Open Challenges and Promising Directions

ML has achieved great success, but the IC design still has some open challenges. In this section, we discuss these open challenges and provide potential solutions from two aspects: scalability and design guidance.

ML training in large-scale IC designs

In modern IC design methodologies, ML-based models are widely used to perform inference since they do not require expensive simulations. Besides, deep learning brings an excellent feature representation to achieve better accuracy. However, industrial designs have become increasingly large, and large-scale IC brings vast challenges. Inputting the entire design will cause out-of-memory issues on GPU. The current solution is partitioning the entire design into several small parts. For example, the layout is partitioned into several clips to input into an ML model for manufacturability verifications [31]. The postlayout netlist is partitioned into several subgraphs to input into an ML model for reliability verification [39]. Then ML-based models replace traditional analytical methods to perform fast simulation and verification. However, this solution cannot capture the long-range contextual information in the layout or netlist. The reliability verification, such as EM verification [72], relies on stress conditions, such as the waveform, voltage, and current in each net. These stress conditions are propagated from the terminals of the design. Currently, these stress conditions have to be obtained by the expensive SPICE simulation. Then each small part with its stress conditions is input into an ML-based model to perform simulation and verification [72]. A potential solution for running time reduction is that the SPICE simulation is performed on a model order reduction [73] circuit to obtain the stress conditions of these small parts, instead of each net. Compared with the original circuit, a model order reduction circuit has less number of nets to achieve speedup in SPICE simulation.

ML to guide IC designs

Once an ML-based model is well trained, it is desired to guide circuit design and physical design and facilitate design closure. We hope to use this model to navigate design so that a high-quality design is efficiently obtained. Many challenges are left, although many frameworks have been proposed to integrate ML-based models into the design stage. For example, for device sizing at the circuit design stage, the design objectives are modeled by black-box GP regression [24], CNN, or GNN models. Other models are adopted as one of the objective functions. However, this strategy cannot reduce search space and search time. As a result, computationally expensive simulation is performed many times to evaluate designs. These models should be treated as constraints in the mathematical formulation to reduce search space and time. This strategy can efficiently select the sequence of inputs for evaluation

to explore high-quality solutions meanwhile satisfying design specifications.

One challenge is integrating an ML-based performance model into the physical design stage to perform performance-aware physical design, for example, placement and routing. Many physical design problems are combinatorial optimization problems, effectively handled by heuristics [74], while gradient-based methods optimize ML-based performance models. Existing ML-based performance-aware physical design schemes rely on specific feature representations (e.g., RUDY) [30] or interaction with an EDA tool [46]. However, it is difficult to model more general performances by the specific feature representations and it is very time-consuming to interact with an EDA tool. The ML-based performance models naturally have the input feature (e.g., layout) gradient, representing a sensitivity of physical design for the specific performance. The critical step is to integrate gradients into existing heuristics. Moreover, it is desired to achieve a significant speedup by the deployment of ML-based performance-aware physical design on high-performance computation platforms, such as GPU.

This Article Summarizes related state-of-the-art research from the taxonomy of ML methodologies, such as shallow models, CNNs, GNNs, generative model, and RL. Moreover, we also discuss many open challenges for scalability and design guidance. We hope this article can be complementary to existing survey articles and promote the development of MLML methodologies in advanced IC designs. ■

Acknowledgments

This work was supported in part by the Research Grants Council of Hong Kong SAR under Grant CUHK14209420, in part by the Innovation and Technology Fund under Grant PRP/065/20FX, in part by the AI Chip Center for Emerging Smart Systems (ACCESS), Hong Kong SAR, and in part by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Project 504518248.

References

[1] M. Rapp et al., "MLCAD: A survey of research in machine learning for CAD keynote paper," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 10, pp. 3162–3181, Oct. 2022.

[2] K. I. Gubbi et al., "Survey of machine learning for electronic design automation," in *Proc. GLSVLSI*, 2022, pp. 513–518.

[3] G. Huang et al., "Machine learning for electronic design automation: A survey," *ACM Trans. Design Autom. Electron. Syst.*, vol. 26, no. 5, pp. 1–46, Jun. 2021.

[4] B. Khailany et al., "Accelerating chip design with machine learning," *IEEE Micro*, vol. 40, no. 6, pp. 23–32, Nov. 2020.

[5] C. M. Bishop, *Pattern Recognition and Machine Learning*, vol. 4. New York, NY, USA: Springer, 2006.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[8] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NIPS*, 2017, pp. 1024–1034.

[9] D. P. Kingma and M. Welling, "Auto-encoding variational Bayes," 2013, *arXiv:1312.6114*.

[10] I. Goodfellow et al., "Generative adversarial nets," in *Proc. NIPS*, vol. 27, 2014, pp. 1–9.

[11] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[12] A. B. Kahng et al., "'Unobserved corner' prediction: Reducing timing analysis effort for faster design convergence in advanced-node design," in *Proc. DATE*, 2019, pp. 168–173.

[13] W. Ye et al., "Tackling signal electromigration with learning-based detection and multistage mitigation," in *Proc. ASPDAC*, Jan. 2019, pp. 167–172.

[14] B. K. Joardar, T. K. Bletsch, and K. Chakrabarty, "Learning to mitigate Rowhammer attacks," in *Proc. DATE*, Mar. 2022, pp. 564–567.

[15] S. Mirbagher-Ajorpaz et al., "PerSpectron: Detecting invariant footprints of microarchitectural attacks with perceptron," in *Proc. MICRO*, Oct. 2020, pp. 1124–1137.

[16] M. Clark et al., "DozzNoC: Reducing static and dynamic energy in NoCs with low-latency voltage regulators using machine learning," in *Proc. IPDPS*, May 2020, pp. 1–11.

[17] B. Donyanavard et al., "SPARTA: Runtime task allocation for energy efficient heterogeneous many-cores," in *Proc. CODES+ISSS*, Oct. 2016, pp. 1–10.

[18] S. Roy et al., "A learning bridge from architectural synthesis to physical design for exploring power efficient high-performance adders," in *Proc. ISLPED*, Jul. 2017, pp. 1–6.

[19] C. Zhuo et al., "From layout to system: Early stage power delivery and architecture co-exploration," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 7, pp. 1291–1304, Jul. 2019.

[20] Q. Sun et al., "Correlated multi-objective multi-fidelity optimization for HLS directives design," *ACM Trans. Design Autom. Electron. Syst.*, vol. 27, no. 4, pp. 1–27, Jul. 2022.

[21] N. Bellarmino et al., "Exploiting active learning for microcontroller performance prediction," in *Proc. ETS*, 2021, pp. 1–4.

[22] F. Last and U. Schlichtmann, "Feeding hungry models less: Deep transfer learning for embedded memory PPA models: Special session," in *Proc. MLCAD*, Aug. 2021, pp. 1–6.

[23] J. Huang et al., "Bayesian optimization approach for analog circuit design using multi-task Gaussian process," in *Proc. ISCAS*, 2021, pp. 1–5.

[24] Q. Sun et al., "Fast and efficient DNN deployment via deep Gaussian transfer learning," in *Proc. ICCV*, 2021, pp. 5380–5390.

[25] B. K. Joardar et al., "Learning-based application-agnostic 3D NoC design for heterogeneous manycore systems," *IEEE Trans. Comput.*, vol. 68, no. 6, pp. 852–866, Jun. 2019.

[26] W.-T.-J. Chan et al., "Routability optimization for industrial designs at sub-14 nm process nodes using machine learning," in *Proc. ISPD*, Mar. 2017, pp. 15–27.

[27] T. Dhar et al., "Fast and efficient constraint evaluation of analog layout using machine learning models," in *Proc. ASPDAC*, 2021, pp. 158–163.

[28] Y. Cao et al., "An efficient and flexible learning framework for dynamic power and thermal co-management," in *Proc. MLCAD*, Nov. 2020, pp. 117–122.

[29] Z. Zhou et al., "Congestion-aware global routing using deep convolutional generative adversarial networks," in *Proc. MLCAD*, 2019, pp. 1–6.

[30] S. Liu et al., "Global placement with deep learning-enabled explicit routability optimization," in *Proc. DATE*, Feb. 2021, pp. 1821–1824.

[31] H. Yang et al., "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.

[32] R. Chen et al., "Faster region-based hotspot detection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 3, pp. 669–680, Mar. 2022.

[33] Z. Xie et al., "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. ASPDAC*, 2020, pp. 13–18.

[34] Z. Xie et al., "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. ICCAD*, 2018, pp. 1–8.

[35] H. Geng et al., "Hotspot detection via attention-based deep layout metric learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 8, pp. 2685–2698, Aug. 2022.

[36] B. Zhu et al., "Hotspot detection via multi-task learning and transformer encoder," in *Proc. ICCAD*, 2021, pp. 1–8.

[37] X. Gao et al., "Layout symmetry annotation for analog circuits with graph neural networks," in *Proc. ASPDAC*, Jan. 2021, pp. 152–157.

[38] K. Kunal et al., "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Proc. DATE*, 2020, pp. 55–60.

[39] T. Chen et al., "Deep H-GCN: Fast analog IC aging-induced degradation estimation," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 7, pp. 1990–2003, Jul. 2022.

[40] Y. Ma et al., "High performance graph convolutional networks with applications in testability analysis," in *Proc. DAC*, 2019, pp. 1–6.

[41] H. Ren et al., "ParaGraph: Layout parasitics and device parameter prediction using graph neural networks," in *Proc. DAC*, Jul. 2020, pp. 1–6.

[42] Z. Xie et al., "Net^2: A graph attention network method customized for pre-placement net length estimation," in *Proc. ASPDAC*, Jan. 2021, pp. 671–677.

[43] Y. Li et al., "A circuit attention network-based actor–critic learning approach to robust analog transistor sizing," in *Proc. MLCAD*, 2021, pp. 1–6.

[44] Y. Li et al., "A customized graph neural network model for guiding analog IC placement," in *Proc. ICCAD*, 2020, pp. 1–9.

[45] S.-C. Hung et al., "Graph neural network-based delay-fault localization for monolithic 3D ICs," in *Proc. DATE*, 2022, pp. 448–453.

[46] K. Zhu et al., "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *Proc. ICCAD*, 2019, pp. 1–8.

[47] Y.-C. Lu et al., "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," in *Proc. ICCAD*, Nov. 2019, pp. 1–8.

[48] B. Xu et al., "WellGAN: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Proc. DAC*, 2019, pp. 1–6.

[49] H. Zhou, W. Jin, and S. X.-D. Tan, "GridNet: Fast data-driven EM-induced IR drop prediction and localized

fixing for on-chip power grid networks," in *Proc. ICCAD*, Nov. 2020, pp. 1–9.

[50] H. Yang et al., "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 10, pp. 2822–2834, Oct. 2020.

[51] G. Chen et al., "DAMO: Deep agile mask optimization for full-chip scale," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 41, no. 9, pp. 3118–3131, Sep. 2022.

[52] J. Liu et al., "Generative adversarial network based scalable on-chip noise sensor placement," in *Proc. SOCC*, Sep. 2017, pp. 239–242.

[53] Z. Chen and D. Marculescu, "Distributed reinforcement learning for power limited many-core system performance optimization," in *Proc. DATE*, 2015, pp. 1521–1526.

[54] Q. Zhang et al., "A double deep *Q*-learning model for energy-efficient edge scheduling," *IEEE Trans. Serv. Comput.*, vol. 12, no. 5, pp. 739–749, Sep. 2019.

[55] S. K. Mandal et al., "An energy-aware online learning framework for resource management in heterogeneous platforms," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 3, pp. 1–26, May 2020.

[56] Z. Zhao and L. Zhang, "Analog integrated circuit topology synthesis with deep reinforcement learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, early access, Feb. 23, 2022, doi: 10.1109/TCAD.2022.3153437.

[57] A. Mirhoseini et al., "Chip placement with deep reinforcement learning," 2020, *arXiv:2004.10746*.

[58] H. Wang et al., "Learning to design circuits," 2018, *arXiv:1812.02734*.

[59] H. Wang et al., "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Proc. DAC*, 2020, pp. 1–6.

[60] A. Hosny et al., "DRiLLS: Deep reinforcement learning for logic synthesis," in *Proc. ASPDAC*, Jan. 2020, pp. 581–586.

[61] A. Agnesina, K. Chang, and S. K. Lim, "VLSI placement parameter optimization using deep reinforcement learning," in *Proc. ICCAD*, Nov. 2020, pp. 1–9.

[62] C. E. Rasmussen and C. K. I. Williams, *Gaussian Process for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[63] B. Scholkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[64] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.

[65] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. CVPR*, Jun. 2015, pp. 3431–3440.

[66] S. Woo et al., "CBAM: Convolutional block attention module," in *Proc. ECCV*, 2018, pp. 3–19.

[67] A. Vaswani et al., "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5998–6008.

[68] P. Veličković et al., "Graph attention networks," in *Proc. ICLR*, 2018, pp. 1–12.

[69] R. Ying et al., "Hierarchical graph representation learning with differentiable pooling," in *Proc. NIPS*, 2018, pp. 4805–4815.

[70] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[71] D. Silver et al., "Deterministic policy gradient algorithms," in *Proc. ICML*, 2014, pp. 387–395.

[72] W. Jin et al., "EMGraph: Fast learning-based electromigration analysis for multi-segment interconnect using graph convolution networks," in *Proc. DAC*, Dec. 2021, pp. 919–924.

[73] A. Odabasioglu, M. Celik, and L. T. Pileggi, "PRIMA: Passive reduced-order interconnect macromodeling algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 17, no. 8, pp. 645–654, 1998.

[74] C. J. Alpert, D. P. Mehta, and S. S. Sapatnekar, *Handbook of Algorithms for Physical Design Automation*. Boca Raton, FL, USA: CRC Press, 2008.

Tinghuan Chen is a postdoctoral fellow at The Chinese University of Hong Kong (CUHK), Hong Kong. His research interests include machine learning in electronic design automation and analog/mixed-signal VLSI design for reliability. Chen has a PhD from CUHK. He is a Member of IEEE.

Grace Li Zhang is a postdoctoral researcher in the Chair of Electronic Design Automation at Technical University of Munich (TUM), 80333 Munich, Germany. Her research interests include high-performance and low-power design, as well as emerging systems. Zhang has a Dr.-Ing. from TUM.

Bei Yu is an associate professor in the Department of Computer Science and Engineering at The Chinese University of Hong Kong, Hong Kong. His research interests include machine learning with applications in electronic design automation and computer vision. Yu

has a PhD in electrical and computer engineering from The University of Texas at Austin, Austin, TX, USA. He is a Senior Member of IEEE.

Bing Li is a group leader in the Chair of Electronic Design Automation at Technical University of Munich (TUM), 80333 Munich, Germany. His research interests include machine learning for electronic design automation and computer architectures. Li has a Dr.-Ing. from TUM. He is a Senior Member of IEEE.

Ulf Schlichtmann is a professor and the head of the Chair of Electronic Design Automation, Technical University of Munich (TUM), 80333 Munich, Germany. His research interests include electronic design automation for designing reliable and robust systems. Schlichtmann has a Dr.-Ing. in electrical engineering and information technology from TUM. He is a Senior Member of IEEE.

■ Direct questions and comments about this article to Bei Yu, Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong; byu@cse.cuhk.edu.hk.