Appendix 9

9.1 Machine Learning in Physical Design

9.1.1 Introduction

In the present late-CMOS era, IC physical design faces three intertwined challenges: cost, quality, and predictability. Cost corresponds to engineering effort, compute effort, and schedule. Quality corresponds to power, performance, and area (PPA) competitive metrics, along with other criteria such as reliability and yield which also determine cost. Predictability corresponds to the reliability of the design schedule, e.g., whether there will be unforeseen floorplan iterations, or whether detailed routing or timing closure flow stages will have larger than anticipated runtimes. The quality of results (QoR) must also be predictable. Each of these three challenges provides a corresponding lever for scaling. In other words, reduction of design cost, improvement of design quality, and reduction of design schedule are all forms of design-based equivalent scaling [7] that expand the availability of leading-edge technology to designers and their IC products. Today, the IC industry looks to machine learning (ML) to provide such benefits across EDA tools, flows, and design methodologies.

In this Sect. 9.2, we first review the promise and challenges for ML in IC physical design, and then illustrate benefits to schedule and quality of results, as demonstrated in recent publications. Among canonical ML applications in physical design is the removal of unnecessary design and modeling margins through learning-based correlation mechanisms. Another example is achieving faster design convergence through predictors of downstream flow outcomes. In addition to applying ML to individual physical design tasks, physical design methodologies can be adapted to benefit from ML techniques and to make these techniques successful. We point out available surveys on ML in physical design, and review ML-based methods for tasks addressed in the preceding chapters. We also attempt to extrapolate recent progress and forecast improvements that appear likely in the future.

A. B. Kahng, J. Lienig, I. L. Markov, J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer 2022, https://doi.org/10.1007/978-3-030-96415-3_9

9.1.2 ML: Promise and Challenges in Physical Design

To motivate specific applications of ML in physical design, we first review the rationale for applying ML techniques and describe their common limitations in the context of physical design. High-performance ML techniques were developed to make use of large amounts of data, e.g., find correlations between signals, solve approximate classification tasks, and compute numerical predictions with a good degree of accuracy. In particular, deep learning excels at combining large amounts of data in structured ways (feature identification) to make predictions. Generative ML techniques are also available, but have historically struggled with combinatorial optimization. In contrast, past physical design R&D focused on optimization problems with serious efforts made to scale to large data sets. Predicting post-optimization quality metrics has always been a challenge. This complementarity in areas of strength is key to understanding recent and emerging applications of ML methods to physical design.

To develop uses of ML in physical design, one typically:

- Chooses impactful physical design tasks that play to the strengths of ML techniques and help amplify existing physical design tools
- Ensures that adequate and sufficient data are available to train and evaluate ML models
- Clarifies optimization objectives and ML-centric loss functions, as well as the tolerance for mistakes in specific tasks
- Defines a runtime regime compatible with the ML techniques involved

In the following, we illustrate how these principles are reduced to application.

9.1.3 Canonical ML Applications

Several ML uses in physical design have become canonical in the past decade.

Improving Analysis Correlation Analysis miscorrelations arise when multiple tools return different results for the same analysis task (parasitic extraction, static timing analysis (STA), etc.) on the same input data. Figure 9.1 shows that better accuracy usually requires more computation, making miscorrelations likely between fast estimators and accurate estimators. For example, "sign-off" timing analysis is too expensive for use within tight optimization loops.

Miscorrelation leads to guard bands and/or pessimism in the design flow. For example, if the place-and-route (P&R) tool's timing report determines that an endpoint has positive worst setup slack, while the "golden," sign-off STA tool determines that the same endpoint has negative worst slack, an iteration (ECO fixing step) will be required. On the other hand, if the P&R tool applies pessimism to guard band its miscorrelation to the sign-off tool, this will cause unnecessary sizing,

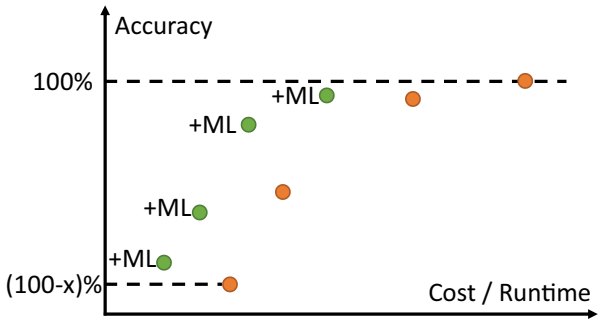

Fig. 9.1 Accuracy-cost trade-off in analysis

shielding, or VT-swapping operations that come at the cost of chip area, power efficiency, and design schedule.

Modeling and Prediction of Tools and Outcomes Convergent, high-quality design requires accurate modeling and prediction of downstream flow steps and outcomes. Predictive models (e.g., of wirelength, congestion, timing) serve as objectives or guides for optimizations. They can also help prevent "doomed runs," thus saving valuable design resources. Without predictive models, the IC physical design process can only "predict by doing," which is akin to predicting a tree's height by watching it grow for 20 years. Ultimately, ML can make physical design faster and more efficient via a "stack" of models that reaches up to the architecture and high-level design layers of chip design.

Several ML directions complement modeling and prediction efforts. Feature identification identifies the structural attributes of design problems—such as size, fanout, utilization, or clock frequency—that determine flow outcomes. Clustering of circuit netlists, often after embedding into vector spaces, helps to identify "natural structure" that can be preserved in layout solutions while also reducing the complexity of partitioning, floorplanning, and placement problems. Developing synthetic designs that are realistic from the standpoint of physical design optimizations can improve the robustness of ML models. More broadly, tool and flow predictions will increasingly span multiple design steps: the analogy is that we must predict what happens at the end of a longer and longer rope when the rope is wiggled.

Other Applications ML methods are also applied within physical design algorithms, tools, and flows in the following two ways: (1) Reinforcement learning and model-guided optimization can apply to small-scale optimizations, such as sizing in timing closure or incremental place-and-route to repair DRC violations. (2) Distributed or federated learning and optimization, as well as adaptive sampling strategies, can apply at the tool and flow levels, especially with the availability of cloud computing resources. This includes evolutionary methods and hyperparameter optimization ("autotuning").

Physical Design Methodologies Can Leverage Machine Learning Physical design methodologies can leverage machine learning by using more accurate predictions to reduce iterations and by adopting ML-powered tools, but they also need to provide sufficient data, metrics, and criteria for success to support ML techniques. In particular, the number of large modern IC designs available for training ML models to any one developer is limited, and new designs sometimes depart from old designs in component use, style, and overall structure. Tasks dealing with polygonal layouts are better suited to address this data challenge, but the general approach is to train ML models on designs and/or layouts that are representative of future inputs, with or without controlled data augmentation. In using this approach, one must be careful to avoid data leakage, i.e., not to train ML models on problem instances too similar to those used in evaluation. Aside from misleading evaluation, such missteps may lead to overfitting, where performance on new inputs lags behind that on previously seen inputs. Another potential disconnect may occur between task-specific metrics and formal ML objectives. To ensure a better match, one can leverage available flexibility in formal ML objectives via hyperparameter tuning, but in some cases better objectives can be extrapolated from feedback, e.g., via reinforcement learning. Given that ML-based methods produce some fraction of less-than-ideal results, one must be clear on how such results can be tolerated and how good results are leveraged. Yet another challenge for methodologies in applying ML techniques is justifying the necessary runtime for a given context. ML techniques vary in their runtimes, but can often be sped up by early termination, data sampling or clustering, as well as hardware acceleration.

ML for IC physical design can also be made more efficient by (1) standardized tool metrics format and (2) data augmentation. Standardized metrics format reduces unnecessary efforts to build ad hoc approaches to design, tool, and flow data collection. Moreover, it also facilitates sharing of ML models and reproduction of experiments [13]. In addition, the lack of data for ML training due to the limited supply of open IC designs can be resolved by data augmentation. Data augmentation can improve the generality of ML models by covering outliers from "unseen" designs.

9.1.4 The State of ML for Physical Design

The remainder of this Sect. 9.1 reviews promising uses of ML, mapped to chapter topics in this book. Additionally, five works that give a broader sense of the scope and velocity of ML for physical design are the following:

- Yu et al. [3] survey common ML and pattern matching techniques, along with applications in physical design and verification such as lithography hotspot detection, datapath placement, and clock optimization.

- Kahng [1, 2] reviews challenges and opportunities for machine learning, along with concrete existence proofs, for application of machine learning in IC physical design.
- Huang et al. [4] give a comprehensive review of existing ML for EDA studies, organized following the EDA hierarchy; this review includes numerous works in the realm of physical design.
- Pandey [5] describes how ML techniques enable the development of next-generation EDA tools with substantial gains in performance and ease of use.
- Rapp et al. [6] present a comprehensive categorization of how ML may be used for design-time and runtime optimizations and exploration strategies in IC design.

Chapter 3: Chip Planning

Among the major stages of chip planning—floorplanning, pin assignment, and power planning—ML has been most actively applied to the macro-block placement problem in floorplanning. Floorplanning is typically formulated as minimization of a combination of wirelength and area, with penalization of overlaps. Over the span of decades, numerous works apply simulated annealing to slicing tree, sequence pair, or B*-tree floorplan representations. More recently, deep Q-learning (DQN) is proposed to select a candidate neighbor solution at each step during simulated annealing. Mirhoseini et al. [18] pose macro-placement as a reinforcement learning (RL) problem and train an agent (i.e., a policy network) to place the nodes of a chip netlist onto a chip canvas. In each iteration of training, all of the macros of the chip block are sequentially placed by the RL agent, after which the standard cells are placed by a force-directed method.

For power planning, Chhabria et al. [11] apply ML to create a power delivery network (PDN) with region-wise uniform pitches based on a library of predesigned, stitchable power grid templates. The methodology is applicable at both the floorplan and placement stages of physical design. At the floorplan stage, an optimized PDN is synthesized based on the early estimates of current and congestion, using a multi-layer perceptron classifier. At the placement stage, an existing PDN is improved based on more detailed congestion and current distributions, using a CNN. At either stage, the neural network builds a safe-by-construction PDN that meets IR drop and electromigration (EM) robustness requirements.

Chapter 4: Global and Detailed Placement

Recall that after chip planning, placement seeks to determine the locations of standard cells or other logic elements within a given layout region while addressing optimization objectives such as minimization of total half-perimeter wirelength. In the placement realm, ML has been applied particularly to global placement.

Works that apply ML in the context of commercial EDA include PL-GNN [17], a graph learning-based framework that provides placement guidance for commercial placers by generating cell clusters based on logical affinity and manually defined attributes of design instances. For a given netlist, the PL-GNN first performs unsupervised node representation learning using graph neural networks (GNNs) based on the initial features manually defined for each design instance, and then

applies weighted K-means clustering to group instances into different clusters. Agnesina et al. [8] use deep RL to automatically optimize runtime parameters of a commercial placement tool, using a mixture of handcrafted topological features along with graph embeddings generated using unsupervised graph neural networks (GNNs). Compared to a state-of-the-art, multiarmed bandit-based tool autotuner, the RL framework is able to consider input netlist features and accurately predict wirelength for unseen netlists based on just one placement iteration.

The stochastic gradient descent optimization method, in conjunction with highly capable hardware platforms for ML, has also led to placement advances. Notably, Lin et al. [15] have developed a novel GPU-accelerated placement framework, DREAMPlace, which casts the analytical placement problem equivalently to training a neural network.

Chapter 5: Global Routing

In the context of global routing, ML is used to supplement existing methods that predict overflows and DRC hotspots based on placement information. Such predictions must be accurate and fast, so that the global routing solution gives viable guidance to the much more time-consuming detailed routing. A variety of ML methods for the global routing stage of physical design have been proposed for both FPGA and ASIC tools.

Due to the highly structured, inhomogeneous nature of FPGA physical layout resources, ML to improve and accelerate FPGA physical design has seen strong activity in recent years. Recently proposed congestion estimation methodologies for FPGA placement include surrogate ML models using a number of alternative classifiers, and conditional generative adversarial network (CGAN)-based modeling.

In the ASIC context, congestion and DRC predictors have used classical regression and image-based ML methods. Multivariate adaptive regression splines (MARS) is an effective regression technique when the correlation between input variables and output results has piecewise linear characteristics. ML applications in the global routing stage also include CNN-based DRC hotspot prediction to forecast DRC hotspots. For example, *RouteNet* [19] predicts the number of design rule violations and detects DRC hotspots by using CNN and fully convolutional network (FCN) models.

While most routing congestion predictors assume availability of placement information, having a predicted congestion map earlier in physical design can help the designer run routability-aware placement and reduce turnaround time. Methods at the preplacement stage include graph attention network (GAT)-based routing hotspot prediction, and wirelength estimation model selection through linear discriminant analysis.

Chapter 6: Detailed Routing

Detailed routing is the most time-consuming of all phases of physical design. Hence, even ML models with high-cost input data (e.g., obtained post-global routing) can have utility if they can achieve accurate detailed routing predictions.

The large amount of labeled data required for training of supervised learning is considered as a barrier to apply ML to EDA. Gandhi et al. [12] propose a data-independent reinforcement learning-based routing model (Alpha-PD-Router) to route a circuit and correct short violations. Alpha-PD-Router uses a collaborative min-max game framework and standard routing algorithms. The framework considers two game players, a path search algorithm-based *Router* and a *Cleaner*, which, respectively, discover and fix design rule violations. The players in turn find and rip up the best net to fix the violation, and then reroute the net.

As noted above, ML-based predictions can also inform optimizations. Chan et al. [10] propose a machine learning framework to *optimize* detailed routing DRC violations. The framework includes a post-global routing DRC violation predictor and a detailed placement optimizer that reduces detailed routing DRCs through predictor-guided cell spreading. This framework is notable both for its predictions (such as hotspot and quality of results) and for performing actual optimization based on prediction data.

Chapter 7: Specialized Routing

On-chip clock distribution solutions are increasingly critical to achieving IC power, area, and design quality goals due to the scaling of technology nodes. In parallel with the improvement of classical algorithms, researchers have used ML to obtain better results from existing tools, and to design new algorithms.

To model existing tools, an artificial neural network (ANN) can be used to estimate the number of clock buffers, and can be applied to each clock gate as well as the clock source in an ideal clock network [14]. The clock structure is then constructed using such estimated clock buffers. A minimized number of buffers can be found by binary search, where at each step the trained ANN is used to find clock parameters for a target number of buffers.

Novel clock tree synthesis (CTS) algorithms based on machine learning include GAN-CTS [16], which uses a conditional generative adversarial network and reinforcement learning to predict and optimize clock tree synthesis outcomes.

Chapter 8: Timing Closure

The physical design of an integrated circuit must satisfy not only geometric requirements, such as nonoverlapping cell placement or design rule correct routing, but timing and other electrical constraints as well. Timing closure, the optimization process that satisfies these constraints, is based on sign-off-accurate timing analysis that is often slow and expensive. Thus, numerous ML approaches have aimed to reduce the runtime burden of high-accuracy static timing analysis (STA) and shift the accuracy-cost trade-off of Fig. 9.1.

All of these works can be classified into two categories: (1) predicting the timing reports of runtime-expensive but accurate tools or models, based on the results of inaccurate but fast tools or models, and (2) estimating the timing information at an

early stage of the IC design flow. Examples in the first category include correcting divergence between two arbitrary timing tools with respect to flip-flop setup time, cell arc delay, wire delay, stage delay, and path slack at timing endpoints; mapping the results of a tool-internal incremental STA engine to that of a sign-off STA tool; predicting timing reports in signal integrity (cross talk-aware) mode, based on timing reports from non-SI mode; predicting expensive path-based analysis (PBA) results from relatively inexpensive graph-based analysis (GBA) results; and predicting the timing analysis at unobserved corners from analysis results at observed corners. Works in the second category mainly focus on estimating the post-placement or post-routing timing information before placement or routing. For example, Barboza et al. [9] propose an ML-based pre-routing timing prediction approach.

9.1.5 Future Developments

Recent and ongoing efforts have improved physical design methodologies with ML techniques, particularly in analysis correlations as well as guiding and tuning traditional optimization methods. Future wins can be gleaned from recent research publications and are likely to be supported in several ways:

- Improved data efficiency, e.g., better feature selection, structural data augmentation, and data-efficient ML models
- Learning to learn, especially learning appropriate optimization objectives and tuning available optimizers to special cases
- More judicious use of "heavier" ML techniques that previously required prohibitive computational resources, including transformer for spatial data, graph neural networks, and reinforcement learning
- Emerging ML techniques for solving combinatorial optimization problems
- Multistage ML techniques that employ lightweight methods for easier tasks and heavier methods for harder tasks, all within an architecture that pursues a higher level goal

A. B. Kahng, J. Lienig, I. L. Markov, J. Hu, *VLSI Physical Design: From Graph Partitioning to Timing Closure*, Springer 2022, https://doi.org/10.1007/978-3-030-96415-3_9