Accelerating Chip Design With Machine Learning

Brucek Khailany, Haoxing Ren, Steve Dai, Saad Godil, Ben Keller, Robert Kirby, Alicia Klinefelter, Rangharajan Venkatesan, Yanqing Zhang, Bryan Catanzaro, and William J. Dally
NVIDIA Corporation

***Abstract*—Recent advancements in machine learning provide an opportunity to transform chip design workflows. We review recent research applying techniques such as deep convolutional neural networks and graph-based neural networks in the areas of automatic design space exploration, power analysis, VLSI physical design, and analog design. We also present a future vision of an AI-assisted automated chip design workflow to aid designer productivity and automate optimization tasks.**

As Moore's law has provided an exponential increase in chip transistor density, the unique features we can now include in large chips are no longer predominantly limited by area constraints. Instead, new capabilities are increasingly limited by the engineering effort associated with digital design, verification, and implementation. As applications demand more performance and energy efficiency from specialization in the post-Moore's-law era, we expect required complexity and design effort to increase.

Digital Object Identifier 10.1109/MM.2020.3026231
Date of publication 24 September 2020; date of current version 21 October 2020.

Historically, these challenges have been met through levels of abstraction and automation. Over the last few decades, electronic design automation (EDA) algorithms and methodologies were developed for all aspects of chip design—design verification and simulation, logic synthesis, place-and-route, and timing and physical signoff analysis. With each increase in automation, total work per chip has increased, but more work was offloaded from manual effort to software. With machine learning (ML) transforming software in many domains, we envision this trend continuing with ML-based automation of EDA.

In this article, we highlight selected work from our research group and the community applying ML to chip design tasks. We also present a vision

Published by the IEEE Computer Society

for a future AI-assisted design flow, where GPU acceleration, neural-network predictors, and reinforcement learning techniques combine to automate the VLSI design.

ML OPPORTUNITY

DL has enabled tremendous advances in many application areas, including computer vision, image processing, and natural language processing. Key drivers for these successes include large available datasets for training, large and increasing model sizes, and accelerated computing platforms such as GPUs. With ML, rather than implementing hand-tuned algorithms and heuristics for each application area, researchers train models using general-purpose techniques to learn application-specific functions directly from millions of examples. The goal is to ultimately deploy a trained model in a production flow to make accurate inferences on previously unseen data.

In this article, we highlight selected work from our research group and the community applying ML to chip design tasks. We also present a vision for a future AI-assisted design flow, where GPU acceleration, neural-network predictors, and reinforcement learning techniques combine to automate the VLSI design.

Learning Techniques

Supervised learning is commonly used in computer vision and other fields. Researchers train a model using many thousands to millions of examples of labeled data from a curated dataset. A trained model can later be used to make an inference on new data. Depending on the application, training labels can come from previous work, human annotations, or ground-truth simulator data. The model's inference accuracy ultimately depends on the quality of the training data, so significant attention must be given to curation of high-quality labeled datasets. In EDA, supervised learning has often been proposed as a fast approximation for complicated, time-consuming tools. However, training data collection is often a challenge. Although many EDA tools produce a lot of data, it is not always labeled or must be curated to be used with DL. For example, data imbalance can be a problem since labeled outlier examples can easily be overlooked with common training methods. In the absence of preexisting labeled training data, an alternative is to use unsupervised learning techniques.

One traditional ML technique familiar to the EDA community for optimization is autotuning methods such as Bayesian Optimization.[1] It is effective when the parameter space is small and there is less benefit from transferring learning from previous experience. With more complicated problems, methods such as deep reinforcement learning (DRL) can be used. DRL selects actions to perform in an environment. Training data are generated on the fly through trial and error as reward functions from the environment classify results of the actions. With enough compute resources, DRL methods can scale to handle a large space of potential actions. Furthermore, with a transferable model architecture, DRL can learn weights during training that can enable direct inference or fast retraining when a model is applied to a new environment or a new design. DRL has already been applied to various games to achieve super-human performance levels.[2] These games have key attributes that make them suitable for DRL: they are discrete, deterministic, fully observable Markov decision processes from which it is easy to gather large amounts of data. Fortunately, many EDA problems also have such attributes, and could be well suited for DRL.

Models

Conventional ML models such as regressions, support vector machines, and gradient-boosted trees[3] have been extensively studied for use in the VLSI design.[4] These models are fast during training and inference, have well-supported software libraries, and are often suitable for simple prediction and correlation tasks.

Recently, more complex DL models based on artificial neural networks trade off more computation for improved accuracy on many classification and regression tasks. They can be trained using backpropagation during supervised or unsupervised learning. DL models can also

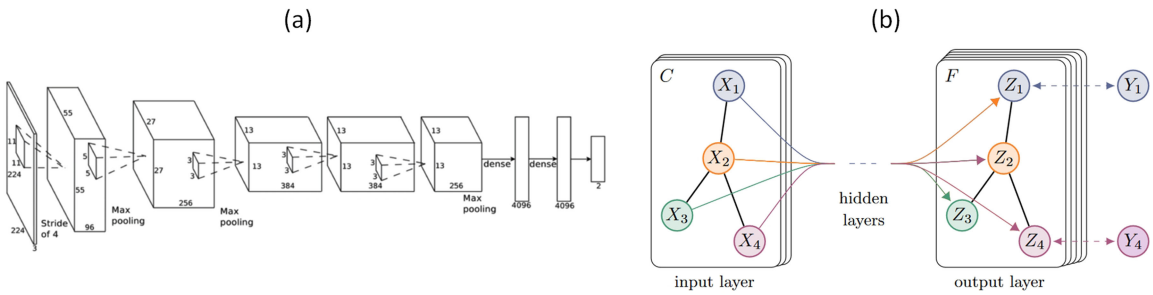

Figure 1. Neural network models. (a) CNN computation in AlexNet.[5] (b) Graph convolutional networks (GCNs).[6]

enable applications that cannot be easily formulated with conventional methods. For example, multilayer perceptrons (MLPs) often consist of several fully connected layers with nonlinear activation functions and can provide better accuracy than traditional ML models on many classification tasks.

Convolutional neural networks (CNNs) such as AlexNet [see Figure 1(a)] are a class of DL models commonly used for image classification or computer vision tasks. Deep CNNs learn to recognize features in 2-D images and to make predictions from learned combinations of image features. Whenever EDA problems can be formulated as structured 2-D data with spatial correlation of features, such as during the VLSI physical design, CNNs are quite suitable. One benefit of CNNs is that many pretrained models are publicly available that can be retrained for EDA tasks. Furthermore, they are fast during training and inference and run efficiently on accelerated computing platforms such as GPUs.

Recently, graph neural networks (GNNs) have emerged as another class of DL models. They are suitable when the data are more naturally represented as a graph than as structured data in a 2-D Euclidian space. One recent example is a graph convolution network [see Figure 1(b)].[6] GNNs are based on two core ideas: graph embedding and neighbor aggregation. Graph embedding learns by transforming graph nodes, edges, subgraphs, and their corresponding features into a lower dimensional vector space representation. Neighbor aggregation learns a node's embedding properties from information on neighboring nodes in the graph. We expect GNNs to be suitable to many EDA problems where data are naturally represented as graphs, such as circuits, logic netlists, or RTL intermediate representations.

ML-BASED PREDICTORS FOR THE CHIP DESIGN

A typical chip design spec-to-layout flow is an iterative process. Large, complex SoCs are split into dozens of units. Each unit proceeds through microarchitecture, RTL design and verification, logic synthesis, timing analysis, power analysis, floorplanning, placement, clock tree synthesis (CTS), routing, and final signoff. Design teams typically overlap work on these steps of the flow by running each of these stages on a different chip revision in a pipelined manner. However, preparing a chip for tapeout still takes many iterations. Each pipeline stage can often take days of manual effort even with EDA tool assistance. Complete iterations through an RTL-to-layout flow often take weeks to months for complex chips.

Trained ML models can help speed time to tapeout by predicting downstream results in a chip design flow, which can reduce time per iteration or improve the quality of results (QoR) such as performance, power, or area. Rather than waiting hours or days for exact results, predictions can be provided in seconds. We highlight four ways predictions can be used: microarchitectural design space exploration, power analysis, VLSI physical design, and analog design.

Microarchitectural Design Space Exploration

Chip designers commonly write parameterized RTL to enable varying product specifications and to allow for microarchitectural tuning by exploring performance, area, and power tradeoffs. With only a few parameters, the design

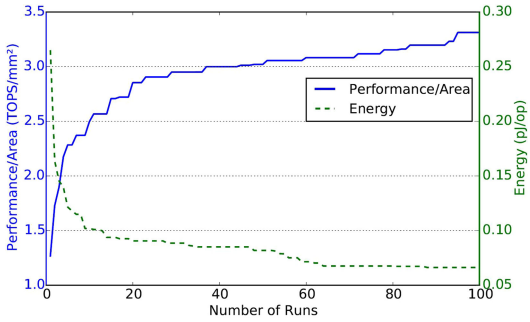

Figure 2. MAGNet tuning via Bayesian optimization.[7]

space can become quite large. For accelerators modeled in higher level RTL generator languages or mapped to RTL through high-level synthesis (HLS), tool knobs can increase potential microarchitectural choices. Navigating this large design space is not always tractable with a brute-force search, especially with limited machine or license resources.

As an alternative approach, we developed MAGNet to automatically search the design space of CNN inference accelerators.[7] It contains 14 tunable design-time parameters controlling various memory sizes, parallel function units, datapath width, datatype precision, and other microarchitectural parameters. The MAGNet flow measures performance by simulating HLS-generated RTL and analyzes power on synthesized gate-level netlists.

MAGNet uses Bayesian optimization (BO) to explore this complex multidimensional design space. As shown in Figure 2, the BO engine starts with random hardware parameters, then iteratively optimizes the objectives (throughput and energy efficiency) of generated designs. The BO engine learns a probabilistic model to approximate the objectives of various parameters based on previously evaluated configurations. In each iteration, new parameters are selected to maximize the expected improvement. The approach finds a desirable energy/performance point in a small number of runs.

More recently, we have found that ML predictors can further improve upon the published MAGNet results by estimating objectives directly from workload characteristics, eliminating many iterations of simulations and logic synthesis. As a proof of concept, we trained three ML models, a linear regression, XGBoost, and a simple MLP with a single hidden layer of 100 neurons, to predict the MAGNet performance and power when running CNN layers from real workloads. Our training dataset was taken from our functional verification tests, readily available from our regression runs, while real workloads were used as the testing set. The input features include the workload specification (CNN layer shapes and sizes), parameters that control the mapping of the CNN layer onto the accelerator, and analytical performance estimates. Features were normalized with a standard scaler. We leveraged a validation set to tune hyperparameters, such as the number and size of MLP layers. The resulting simple models are inexpensive to train and incur negligible runtime overhead during inference.

We tested the models' R^2 accuracy on layers from real image classification CNNs. R^2 is a common metric used to evaluate regression tasks that quantifies the proportion of variance in the prediction. When predicting chip utilization (a measure of performance) directly from the input features, the linear, XGBoost, and MLP models achieve accuracies of 75.1%, 93.8%, and 93.9%, respectively. The models achieve accuracies of 77.3%, 87.5%, and 84.6%, respectively, for predicting power dissipation. The XGBoost and MLP models outperform the linear model due to their handling of nonlinear data. XGBoost is the most accurate power predictor, likely from MLP overfitting on small datasets. Incorporating these predictors into the MAGNet BO framework can enable even faster design space exploration.

Power and IR Drop Analysis

Pre-tapeout power estimation and optimization is a critical aspect of all chip design flows today. Accurate analysis requires running logic simulations on gate-level netlists with annotated capacitive parasitics. However, these simulations are very slow, typically running 10–1000 clock cycles per second, depending on activity factor and design size. To provide faster yet accurate power estimates, ML models can predict dynamic power by estimating the propagation of switching activity factors through a logic netlist without simulation. For example, PRIMAL trained a CNN model to infer power from RTL

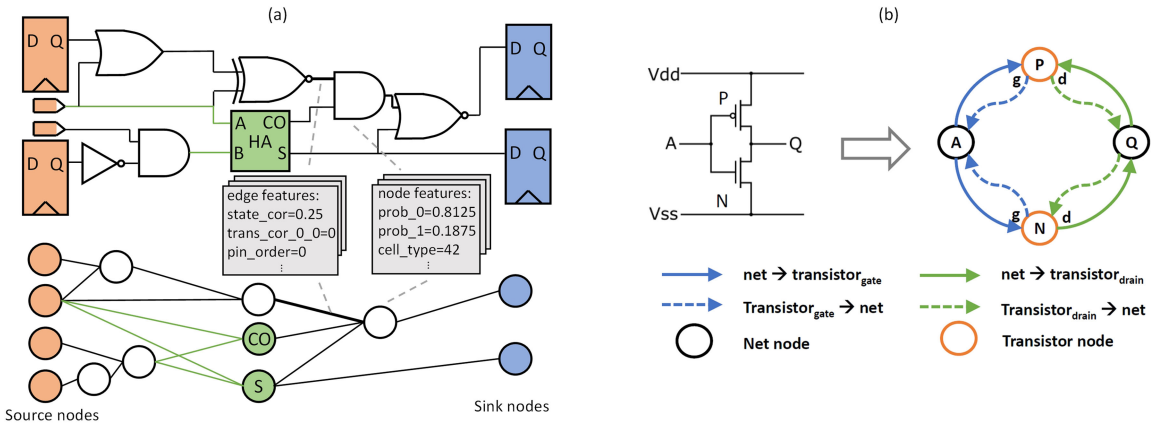

Figure 3. Mapping circuits to GNNs. (a) GRANNITE[9] representation of Boolean logic—logic gates as nodes and wires as edges. (b) ParaGraph[13] representation of an inverter circuit—heterogeneous nodes for transistors and wires, edges for connectivity.

simulation traces.[8] It scaled to large (100k gate) designs at high accuracy, although the trained ML models can only infer power for a new workload on the same design.

GRANNITE[9] builds upon the PRIMAL flow to enable model transferability to new designs with a GNN, shown in Figure 3(a). Gate netlists are translated into graphs with per-node (gate) and per-edge (net) features, such as each gate's intrinsic state probabilities. GNNs can learn from both graph and input activation features, and likewise GRANNITE learns from both RTL simulation trace data and input graph (netlist) data. In this way, the model becomes transferable to both new graphs (netlists) and new workloads. Results show GRANNITE achieving good accuracy (less than 5.5% error across a diverse set of benchmarks) for fast (<1 second) average power estimation on designs up to ~50k gates.

Once designs proceed to physical design, IR drop analysis becomes a critical part of power signoff. Excessive IR drop prevents circuits from running at targeted speeds. Accurate analysis requires solving large systems of linear equations to obtain the voltage of every node in a circuit, which can take days on large designs. Multiple iterations of analysis and mitigation are desirable during a VLSI flow, but conventional analysis is too slow to be used in this way. PowerNet[10] overcomes this challenge by predicting IR drop directly from per-cell power distributions. PowerNet uses each cell's switching power consumption during a timing window as input features. It is trained with supervised learning using ground-truth simulation results as labels to predict the maximum IR drop for each cell across all timing windows. Because its features and model follow first principles from the IR drop analysis, PowerNet can then infer IR drop on designs not seen during training. It is 9% more accurate than the best previously proposed ML method for vectorless IR drop prediction and achieves a 30× speedup compared to a commercial IR drop analysis tool.

Physical Design

Detailed routing is perhaps the most time-consuming stage of a modern physical design flow. It is difficult to determine exactly whether a synthesized or placed design will be DRC violation free while meeting timing constraints after performing detailed routing. However, the earlier synthesis and placement stages have the most leverage on a design's routability. Therefore, it is desirable to predict routability from these early stages. Today's VLSI flows use congestion and other postplacement heuristics to estimate routability. However, they are often inaccurate and require designer intuition to rule out false positives and identify real problems.

Image-based DL models provide an excellent opportunity to predict routability compared to prior heuristic-based approaches. For example, RouteNet[11] leverages a fully convolutional network (FCN) to predict postdetailed-route DRCs from post-placement global routing results.

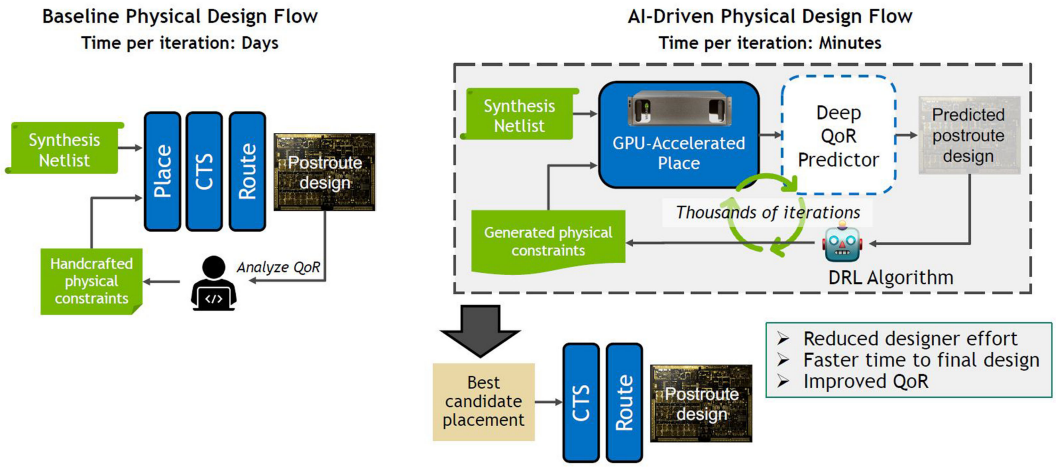

Figure 4. VLSI physical design implementation flows.

Because FCNs can access global information and local window information, their predictions are more accurate than prior techniques based on local windows or congestion heuristics. RouteNet is particularly well suited for designs with macros, which can profoundly impact routability. For DRC hotspot prediction, RouteNet improved accuracy by 50% compared to congestion heuristics and SVM-based methods.

In some cases, routability problems can be identified even earlier in the flow, after logic synthesis. CongestionNet leverages a graph attention network to estimate routing congestion based only on the circuit graph.[12] Its predictions are possible because some routing congestion problems are highly correlated with certain digital circuit topologies, with characteristics that can be recognized by GNNs. Compared to using previous metrics for predicting congestion hotspots without placement information, CongestionNet provides a 29% increase in the Kendall ranking correlation score.

Analog Layout

Analog design, and especially analog layout, is a labor-intensive process still lacking high-quality design automation tools. One difficulty is the larger degree of freedom compared to digital design, which is constrained to standard cell rows. Furthermore, analog circuit QoR metrics are design dependent, whereas digital QoR metrics such as timing, area, and power are universal.

The advancement of AI provides a good opportunity to automate the analog design process. One recent example is ParaGraph,[13] a GNN that predicts layout parasitics and device parameters directly from circuit schematics. Postlayout parasitic prediction is key to automating analog layout generation since it can help with schematic and layout convergence, floorplan feasibility, or QoR estimates. ParaGraph makes accurate predictions by observing that similar circuit topologies and transistor configurations often have similar layouts and therefore similar parasitics. Its GNN architecture uses a heterogeneous graph representation [see Figure 3(b)] incorporating ideas from models such as GraphSage, Relational GCNs, and graph attention networks. Trained on a large dataset of industrial circuits, ParaGraph achieves an average prediction R^2 of 0.772 (110% better than XGBoost) and reduces average simulation errors from over 100% to 10% compared to hand heuristics currently used by designers.

AI-ASSISTED CHIP DESIGN

Augmenting today's semi-automated VLSI flows with ML-based predictors can provide immediate benefits. Looking to the future, we expect even larger breakthroughs to come from deploying AI to directly optimize chip designs without human intervention and to augment or replace existing best-known algorithms in EDA tools. We highlight several promising research directions in these areas.

AI-Driven Floorplanning and Placement

A central challenge in a baseline physical design flow (see Figure 4) is that the backend VLSI tools—placement, CTS, and routing must produce final design-rule and timing clean designs in deeply scaled modern process technologies. As a result, they act as complicated and slow black boxes from synthesized netlists to final routed designs. The tools leave the designer with minimal controllability or observability and (due to time and license constraints) limited opportunity for experimentation. However, the probability of success with good QoR is also highly dependent on hand-crafted physical constraints to the backend flow provided by the designer.

To address this, we envision an AI-driven physical design flow (see Figure 4) to intelligently explore the design space of potential physical floorplans, timing and tool constraints, and placements. This flow can leverage fast GPU-accelerated placers and DL-based deep QoR predictors to enable thousands of fast iterations within the AI-driven loop and avoid costly iterations through downstream black-box tools. By running within a DRL optimization loop, the AI-driven flow can automatically and quickly find high-quality floorplans, timing constraints, and standard cell placements that can achieve good downstream QoR, then dispatch those candidate placements to the downstream CTS and routing steps. This flow is enabled by three key technologies: 1) DRL for physical constraint optimization; 2) fast VLSI placers running on accelerated computing platforms such as GPUs; and 3) DL-based deep QoR predictors to classify postplacement results.

Directly predicting accurate postroute quality from floorplans and timing constraints is difficult. Ideally, we would like to use a fast standard-cell placement engine that can run quickly and be easily integrated into DL training frameworks. Fortunately, recent work has demonstrated this potential with GPU-accelerated VLSI placement.

First, we expect that DRL could explore the search space of physical design knobs (macro placement, pin placement, aspect ratio) and tool settings to find optimal realizations of a given design. For example, recent work showed that DRL with GNNs can improve the QoR of macro-placement over manual design or simulated annealing.[14] With a few hours of training, a trained RL agent produced macro placements with better QoR than the manual design done in weeks. It further has the benefit of learning from previous experience when placing macros in future designs. However, DRL typically requires thousands to millions of trials during training, so it needs fast and accurate reward functions of floorplan quality that are highly predictive of downstream QoR.

Directly predicting accurate postroute quality from floorplans and timing constraints is difficult. Ideally, we would like to use a fast standard-cell placement engine that can run quickly and be easily integrated into DL training frameworks. Fortunately, recent work has demonstrated this potential with GPU-accelerated VLSI placement. DREAMPlace[15] casts the analytical placement problem to be equivalent to training a neural network. Implemented in PyTorch, with customized key kernels for wirelength and density computations, DREAMPlace demonstrated over a 30× speedup without quality degradation compared to a state-of-the-art multi-threaded CPU-based placer. ABCDPlace[16] implements VLSI detailed placement using fast GPU-accelerated graph algorithms and achieves a 16× speedup over a state-of-the-art sequential detailed placer. Together, these placers can place ~1M cell designs in less than 1 min and ~10M cell designs in several minutes. These runtimes are fast enough that many trial placements of synthesized netlists can run with reasonable machine resources as part of a DRL optimization loop.

Finally, DRL needs an accurate QoR predictor: fast reward functions to evaluate candidate placements based on predictions of downstream routability and QoR. In a prior RL-driven macro placement work,[14] proxy wirelength and congestion estimates were used as reward functions. Although such metrics are easy to attain, rewards that better correlate with postroute design quality are preferred. For example, a detailed router may be able to fix DRC errors, but at great runtime and area cost; conversely,

during timing closure steps, tools tend to consume available positive timing slack by downsizing gates or swapping Vt flavors to save power. A more robust algorithm might instead predict *pressure* metrics, heuristics relating to how hard the tool must work to route a group of nets or close timing on a specific path. Designs with high aggregate pressure would likely have poor overall QoR, while localized hotspots of high pressure might result in difficulty with routability or timing closure. It is an open research problem to see if such deep QoR predictors can be trained to evaluate postplacement results. However, if such predictors can be developed and combined with DRL and GPU-accelerated placement, an AI-driven physical design flow could benefit both designer productivity and QoR.

Learning EDA Algorithms With DRL

We also envision that ML can be used to automatically learn new or improve EDA algorithms themselves. Many EDA problems can be formulated as combinatorial optimization problems. Solving them exactly, however, may be intractable. As a result, EDA tools resort to heuristics. In some cases, such as SAT solvers, ML may be able to improve the heuristics. In other cases, DRL may be suitable. For example, logic synthesis engines repeatedly apply transforms such as balancing, rewriting, refactoring, and resubstitution to subcircuits to incrementally improve the QoR. If we treat such transforms as actions, and the circuits after each transform as states, the optimization is a Markov decision process that can be optimized with DRL.

DRL might also be able to reduce runtime for timing optimization in VLSI backend tools. With today's tools, it can take up to a week to run placement, CTS, and routing on designs with millions of cells. Much of that runtime is spent on repeatedly and incrementally optimizing the same cells along with costly static timing analysis (STA) updates. DRL might be able to learn optimal policies that avoid repetitive optimizations on similar cells and reduce the total number of STA updates.

CONCLUSION

Initial research applying ML predictors for VLSI has shown the potential of AI in chip design flows across a variety of tools. In the future, we expect ML-based approaches such as DRL to be suitable for many EDA optimization problems, especially when modeling the exact objective or constraints is difficult. DRL could become a new general-purpose algorithm for EDA, just like simulated annealing, genetic algorithms, and linear/nonlinear programming. Since DL models are optimized to run efficiently on accelerated computing systems such as GPUs, we expect to see a virtuous cycle of exploiting the world's most powerful computers for designing the next generation of chips, which in turn will improve the performance of future EDA algorithms.

REFERENCES

1. J. Mockus, "On bayesian methods for seeking the extremum and their application," in *Proc. IFIP Tech. Conf. Optim. Tech.*., 1977, pp. 400–404.
2. D. Silver *et al.*, "Mastering the game of go without human knowledge," *Nature*, vol. 550, pp. 354–359, 2017.
3. T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.
4. A. B. Kahng, "Machine learning applications in physical design: Recent results and directions," in *Proc. Int. Symp. Physical Des.*, 2018, pp. 68–73.
5. A. Krizhevsky *et al.*, "ImageNet classification with deep convolutional neural networks," in *Proc. NeurIPS*, 2012, pp. 1097–1105.
6. T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016. [Online]. Available: https://arxiv.org/abs/1609.02907
7. R. Venkatesan *et al.*, "MAGNet: A modular accelerator generator for neural networks," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des.*, 2019, pp. 1–8.
8. Y. Zhou *et al.*, "PRIMAL: Power inference using machine learning," in *Proc. 56th Annu. Des. Autom. Conf.*, 2019, Art. no. 39.
9. Y. Zhang *et al.*, "GRANNITE: Graph neural network inference for transferable power estimation," in *Proc. Des. Autom. Conf.*, 2020, pp. 1–6.

10. Z. Xie *et al.*, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Proc. 25th Asia South Pacific Des. Autom. Conf.*, 2020.
11. Z. Xie *et al.*, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. Int. Conf. Comput. Aided Des.*, 2018, pp. 1–8.
12. R. Kirby *et al.*, "CongestionNet: Routing congestion prediction using deep graph neural networks," in *Proc. IFIP/IEEE 27th Int. Conf. Very Large Scale Integr .*, 2019, pp. 217–222.
13. H. Ren *et al.*, "ParaGraph: Layout parasitics and device parameter prediction using graph neural networks," in *Proc. Des. Autom. Conf.*, 2020, pp. 1–6.
14. A. Mirhoseini *et al.*, "Chip placement with deep reinforcement learning," 2020. [Online]. Available: https://arxiv.org/abs/2004.10746
15. Y. Lin *et al.*, "DREAMPlace: Deep learning toolkit-enabled GPU acceleration for modern VLSI placement," in *Proc. 56th Des. Autom. Conf.*, 2019, pp. 1–6.
16. Y. Lin *et al.*, "ABCDPlace: Accelerated batch-based concurrent detailed placement on multi-threaded CPUs and GPUs," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, early access, Feb. 4, 2020, doi: 10.1109/TCAD.2020.2971531.

Brucek Khailany joined NVIDIA in 2009 and is the director of the ASIC and VLSI Research Group. He leads research into innovative design methodologies for IC development, ML and GPU-assisted EDA, and energy-efficient ML accelerators. Khailany received the Ph.D. degree in electrical engineering from Stanford University and the B.S.E. degree from the University of Michigan. He is a senior member of the IEEE. Contact him at bkhailany@nvidia.com.

Haoxing Ren is currently a Principal Research Scientist with NVIDIA. His research interests include machine learning applications in design automation and GPU accelerated EDA. He received many IBM technical achievement rewards including the IBM Corporate Award for his work on improving microprocessor design productivity. He has received the best paper awards at ISPD'13 and DAC'19. Ren received the B.S/M.S. degrees in electrical engineering from Shanghai Jiao Tong University, the M.S. degree in computer engineering from Rensselaer Polytechnic Institute, and the Ph.D. degree in computer engineering from University of Texas at Austin. He is a senior member of the IEEE. Contact him at haoxingr@nvidia.com.

Steve Dai is currently a Research Scientist as part of the ASIC & VLSI Research Group at NVIDIA. His research interests include energy-efficient DL acceleration, high-level design methodologies, and ML-assisted EDA. Dai received the B.S. degree in electrical engineering from the University of California at Los Angeles in 2011, the M.S. degree in electrical engineering from Stanford University in 2013, and the Ph.D. degree in electrical and computer engineering from Cornell University in 2019. Contact him at sdai@nvidia.com.

Saad Godil is the Director of Applied Deep Learning Research at NVIDIA. His research interests include reinforcement learning, graph neural networks, and ML applications for VLSI and Chip Design. Contact him at sgodil@nvidia.com.

Ben Keller joined the ASIC & VLSI Research Group, NVIDIA Corporation, in 2017, where he works as a Senior Research Scientist. His research interests include digital clocking and synchronization techniques, fine-grained adaptive voltage scaling, and hardware design productivity. Keller received the B.S. degree in engineering from Harvey Mudd College in 2010 and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California at Berkeley in 2015 and 2017, respectively. He is an IEEE member. Contact him at benk@nvidia.com.

Robert Kirby is a researcher on the Applied Deep Learning Research team at NVIDIA. His primary research interest is in applying deep learning and reinforcement learning methods to solve challenging engineering problems from across the full semiconductor design flow. Kirby received the B.S. degree in electrical and computer engineering from the University of Illinois at Urbana-Champaign. Contact him at rkirby@nvidia.com.

Alicia Klinefelter joined NVIDIA in January 2017 and is currently a Senior Research Scientist with the ASIC and VLSI Research Group. Her research interests include low-power circuit design, design effort reduction techniques, and emerging verification techniques for high-level languages. Klinefelter received the Ph.D. degree in electrical engineering from the University of Virginia in 2015. She is a member of the IEEE and the Solid-State Circuits Society. Contact her at aklinefelter@nvidia.com.

Rangharajan Venkatesan is a Senior Research Scientist with NVIDIA. His research interests include machine learning accelerators, low-power VLSI design, and SoC design methodologies. He has served as a member of the technical program committees of several leading IEEE conferences including International Solid-State Circuits Conference, International Symposium on Microarchitecture, Design Automation Conference, and International Symposium on Low Power Electronics and Design. Venkatesan received the B.Tech. degree in electronics and communication engineering from the Indian Institute of Technology in 2009 and the Ph.D. degree in electrical and computer engineering from Purdue University in 2014. Contact him at rangharajanv@nvidia.com.

Yanqing Zhang joined NVIDIA, Inc., in January 2014 and is currently a Senior Research Scientist. His research interests include machine learning for EDA applications, digital VLSI methodology, variation resilient digital design, and latch-based timing. Zhang received the Ph.D. degree in electrical engineering from the University of Virginia in 2013. He has been an IEEE member since 2013. Contact him at yanqingz@nvidia.com.

Bryan Catanzaro is VP of Applied Deep Learning Research with NVIDIA. He leads a team that researches in new ways to use deep learning for VLSI and computing system design, computer graphics, natural language understanding, and speech. Catanzaro received the Ph.D. degree in electrical engineering and computer sciences from the University of California, Berkeley. He is a member of IEEE and ACM. Contact him at bcatanzaro@nvidia.com.

William J. Dally is a Chief Scientist and the Senior Vice President of Research with NVIDIA Corporation and a Professor (Research) and the former Chair of Computer Science with Stanford University. He is a member of the National Academy of Engineering, a Fellow of the IEEE, a Fellow of the ACM, and a Fellow of the American Academy of Arts and Sciences. He has received the ACM Eckert-Mauchly Award, the IEEE Seymour Cray Award, the ACM Maurice Wilkes Award, the IEEE-CS Charles Babbage Award, and the IPSJ FUNAI Achievement Award. He currently leads projects on computer architecture, network architecture, circuit design, and programming systems. He has published over 250 papers in these areas, holds over 160 issued patents, and is an author of the textbooks *Digital Design: A Systems Approach*, *Digital Systems Engineering,* and *Principles and Practices of Interconnection Networks*. Contact him at bdally@nvidia.com.