Security and Reliability Challenges in Machine Learning for EDA: Latest Advances

Invited Paper

Zhiyao Xie*, Tao Zhang, Yifeng Peng
Hong Kong University of Science and Technology
*eezhiyao@ust.hk

ABSTRACT

The growing IC complexity has led to a compelling need for design efficiency improvement through new electronic design automation (EDA) methodologies. In recent years, many innovative machine learning (ML)-based solutions have been proposed for EDA applications. While these ML solutions demonstrate great potential in the circuit design flow, however, the hidden security and model reliability problems are rarely discussed until recently. In this paper, we present some latest research advances in the security and reliability challenges in ML for EDA.

1 INTRODUCTION

Driven by the continuously growing complexity of integrated circuits (ICs), design companies are in increasingly greater demand for experienced manpower and stressed with unprecedented longer turnaround time. The nonrecurring engineering (NRE) cost associated with chip design also keeps skyrocketing accordingly [12]. Therefore, there is a compelling need for an essential improvement in IC design efficiency through new methodologies and design automation techniques. To achieve this, machine learning (ML) techniques are considered highly promising.

In recent years, machine learning for EDA has become a trending topic [10, 27]. ML models are developed to improve the predictability in chip design flows, by providing early feedback on downstream design quality or accelerating the solution of EDA problems. These ML models learn from prior design solutions and typically perform orders-of-magnitude faster design quality evaluations or optimizations. We have witnessed ML solutions targeting various design objectives, covering all major design stages for both analog and digital designs [10, 27]. Some techniques are further adopted in commercial EDA tools [2, 30]. In both EDA academia and industry, ML for EDA has made an impressive impact. We have strong reasons to believe ML models will be more widely adopted in design automation in the future.

However, as ML techniques are introduced in design automation, new security and model reliability[1] concerns arise, while many practitioners are not fully aware of them. A most recent survey paper [36] provides the first systematic study on both security and reliability problems in existing ML for EDA solutions. It categorized relevant concerns into four types as below [36].

(1) **Attacks against data privacy**, e.g., attacks that try to infer private information about design data. The victims are the training data providers. The attackers can be malicious competitors targeting access to private data by exploiting their access to trained ML models.
(2) **Attacks against competitive advantage**, e.g., attacks that construct similar substitute ML models, which impair the competitive advantage of the original model. The victims are the model providers who wish to make a profit, and the attacker can be malicious users who try to construct substitute ML models.
(3) **Attacks against ML performance**, e.g., adversarial or poisoning backdoor attacks that cause model accuracy degradation on specific testing samples. Victims are model users, and attackers can be someone who wishes to fool the model and introduce design deficiencies.
(4) **Inherent unreliability in ML performance**, e.g., unexpected serious model accuracy degradation on essentially new test samples. Victims are model users, and there are no attackers in this problem.

The survey [36] has verified the difficulty of attacking data privacy in challenge 1. After its publication, we further achieved some latest advances in all three other challenges 2, 3, and 4. In this paper, we will present some latest research advances about these security and reliability challenges in ML for EDA.

2 ML FOR EDA BACKGROUND

2.1 Existing ML Solutions in EDA

Nowadays, ML-based research efforts can be observed at almost all major stages of a typical VLSI design flow. For high-level synthesis (HLS), models are proposed for fast quality of result (QoR) estimation [6] and design space exploration [16]. Power models are also proposed for fast power estimation [44] or runtime circuit management [33, 38]. At logic synthesis, ML models are proposed for chip quality prediction [42] and optimization [9]. During physical design, models perform predictions or optimizations on almost all important design metrics, including timing [1, 34], macro placement [11, 24], routability [5, 32], IR drop [37, 43], clock tree quality [21], interconnect [35], crosstalk [15], 3D integration [22], etc. Also, ML models are developed for design verification [13], design for testability (DFT) [23], and lithography problems [40]. Besides specific design stages, design flow tuning [31] is another well-explored topic.

ML-based methods are of course not only limited to digital designs. As for analog design, similarly, various models have been

[1]The term *reliability* refers to trustworthy ML outputs, which can be safely used like simulation results without additional time-consuming validation. In this context, this term does **not** refer to the lifespan/robustness of a circuit.

Table 1: Application Scenarios of ML for EDA [36]

Scenario	Black-Box	Trained	Separated Provider & User
S1	✓	✓	✓
S2	✓	✗	✓
S3	✗	✓	✗

developed for topology design [14], device sizing [8], pre-layout prediction [28], layout evaluation [20], layout generation [39], analog design testing [29], etc.

2.2 Application Scenarios

Table 1 presents three potential application scenarios or business models of ML for EDA based on our anticipation. In the first scenario S1, a separate ML model provider provides their well-trained model as a black box to users, possibly through cloud services. This is very similar to the popular ML-as-a-service (MLaaS) business model in many general ML tasks, like the cloud services offered by Amazon, Google, Microsoft, BigML, etc. Such cloud services allow model providers to charge users for queries. These ML models are of high commercial value. In this case, models will be vulnerable to attacks against competitive advantage, attacks against ML performance, and also unreliability problems.

Another possible scenario, S2, is to leave more tasks to users. The model providers only design their ML methodology without performing the training. The method is provided as a black box, with information like selected features, model architecture, and the optimization procedure not explicitly disclosed. Then users can train and use their own customized ML models as a black box with their own labeled data. Rather than being provided as standalone services in S1, it is more likely for such methodologies to be integrated and released together with existing EDA tools. This business model can already be observed in some existing EDA tools from vendors. In this case, models will be vulnerable to attacks against ML performance and also unreliability problems.

Finally, ML model providers and users may not be separated. Users in design companies can design and train their own models for specific problems in their in-house design flow. This is scenario S3 in Table 1. This rather private flow is less vulnerable to malicious attacks. But it will still be affected by the inherent unreliability of some ML models.

3 ATTACKS AGAINST ML MODEL COMPETITIVENESS

3.1 ML Models Competitiveness Overview

As indicated by scenario S1 in Table 1, trained ML models can be provided on the cloud as a service in ML for EDA. Such MLaaS typically charges clients based on their queries. For service providers, it takes extensive efforts to construct high-quality models, involving data collection, label generation, ML model design, ML model training and testing, etc. Considering such costly development process, these trained ML models are important business asset.

However, it is possible for attackers to 'steal' these models. Here the 'steal' broadly refers to all activities where attackers develop their own substitute ML models with very similar functionality, utilizing the existing model in MLaaS. In other words, based on an existing black-box model F, attackers can train their own model, named *attack model* F_a, with a much lower cost. This malicious attack is also referred to as *model extraction*. Although this attack does not affect the function of the original MLaaS, the attack model F_a poses an obvious threat to the competitive advantage and business value of the original model F. But this potential threat to model providers is rarely studied until recently.

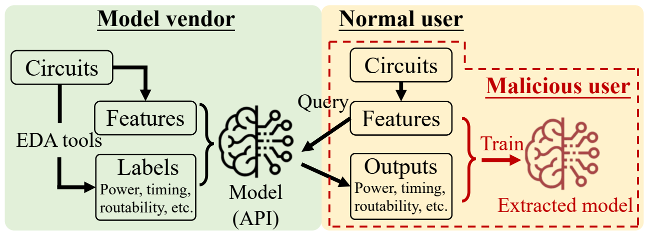


Figure 1: An anticipated application scenario of ML for EDA model and the corresponding model extraction attack. The vendor trains a model with circuit features and labels generated by EDA tools. Users are expected to take their own circuit features as inputs to query their circuits' quality. However, malicious users can take advantage of model outputs to construct similar substitute models, without the costly label generation process [3].

3.2 Attack Method on Model Competitiveness

For attackers who hope to build their own attack model F_a in scenario S1, they can actually greatly benefit from an existing trained ML model by accessing it as a black box. Our most recent work [3] studies this attack and validates that it can jeopardize the development and commercialization of ML models for EDA in scenario S1. Fig. 1 shows the anticipated scenario of this attack. Attackers can query the existing ML model with their own circuit features to generate predicted circuits' quality. Attackers then train their own substitute model using predictions as pseudo-labels. In this way, attackers skip the label generation process, which is the most time-consuming step in the development of ML models in EDA, since this requires extensive usage of EDA tools. By skipping the label generation, the model construction process will be much easier for malicious attackers.

In this scenario, the attacker has two goals: 1) Maximize the accuracy of his own substitute ML model. Ideally, the accuracy of this substitute/extracted model should approach the target victim model. However, since the victim model may produce false pseudo-labels due to the model's imperfection for unseen data, it could actually mislead the attack model. Therefore, an accuracy gap is expected. 2) Minimize the number of queries to the victim model during the training process. In practice, model vendors (i.e. the victim) may charge users per query or set a hard limit to prevent potential attacks. Therefore, minimizing queries leads to a higher attack efficiency.

When the number of queries is restricted, querying the victim model with appropriate data is essential to the attacker model's final performance. This work [3] proposes an information-based iterative data selection method to progressively select data that could give the most training enhancement to the attacker model.

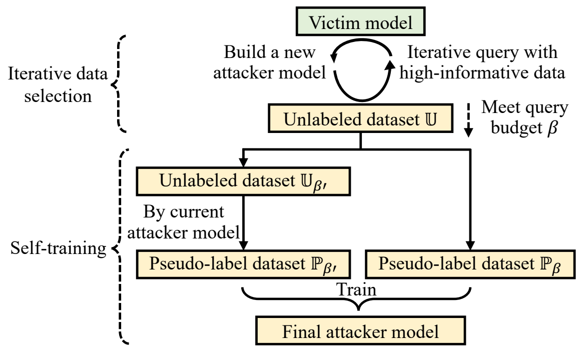

Victim model
Build a new attacker model
Iterative query with high-informative data
Iterative data selection
Unlabeled dataset $\mathbb{U}$
Meet query budget β
Unlabeled dataset $\mathbb{U}_{\beta'}$
By current attacker model
Self-training
Pseudo-label dataset $\mathbb{P}_{\beta'}$
Pseudo-label dataset $\mathbb{P}_{\beta}$
Train
Final attacker model

Figure 2: Overview of the information-based iterative data selection method. The attacker iteratively queries the victim model by high-informative unlabeled data according to the current attacker model. After meeting the query budget, the attacker uses the current model to label the rest data and train the final model with the whole pseudo-label dataset [3].

This attack method is shown in Fig. 2 [3]. In each iteration, it performs two major operations: 1) iterative data selection, 2) self-training method. Iterative data selection progressively selects high-informative data corresponding to the attack model in the current iteration until meeting the query budget. Then self-training method updates the attack model by training with both victim-model-labeled data and unlabeled data.

The method first queries the victim model with random samples from different circuits and uses their pseudo labels to train an initial attack model. Based on this attack model, it enters the loop of iterative data selection and model updating. In each iteration, for each unqueried data sample, it estimates its amount of information by the entropy in the prediction from the attack model. A prediction with high entropy reflects the attack model's uncertainty on this prediction. That is, the model lacks knowledge of this input sample, and thus pseudo-labeling this sample can potentially maximally enhance the attack model. This policy is similar to the concept of active learning.

3.3 Experiment on Model Competitiveness Attack

The attack method is evaluated with the routability prediction task. ML models detect the locations of DRC violations before routing. Table 2 shows the accuracy of the original victim model and the substitute attack model. This experiment assumes the attacker does not know the actual model architecture of the victim model. According to Table 2, the accuracy of the attack model can get very close to or even slightly better than the victim model. Remember that the attacker does not need to generate any label. It indicates an obvious threat of the attack on model competitiveness. More detailed results about the sampling algorithm's efficiency in this attack are given in the original paper [3].

4 ATTACKS AGAINST ML PERFORMANCE

4.1 ML Performance Attack Overview

Many other malicious attacks target affecting the performance of existing ML models. Some prior works [17, 19, 41] studied the attack on the performance of CNN-based lithography hotspot detectors.

Table 2: Routability prediction results with unlimited access to victim model [3]. Since attacker may not know the victim ML model architecture, two different ML models [4, 32] are applied for victim and attacker.

Models & Methods	ROC-AUC	ROC-AUC
Victim	0.892 [32]	0.871 [4]
Proposed attack method [3]	**0.880** [4]	**0.875** [32]

There are multiple types of malicious attacks on the performance of ML models. A well-studied type is adversarial attack, where attackers modify the model input by very small but deliberate alterations, named adversarial perturbation. In this way, attackers introduce their desired misleading ML inference result, without being noticed by potential victims.

The work of [19] presents a realistic scenario of adversarial attacks on ML models targeting lithography hotspot detection. Currently, using a CNN-based hotspot detector, the designer can quickly ascertain if a layout with third-party macros is printable. To pass off sub-par designs as high quality, malicious third-party vendors may selectively modify their layouts to steer the detector to misclassify hotspot regions as non-hotspot. That is, attackers can hide hotspots in their low-quality macros by introducing adversarial perturbations.

Besides adversarial attacks, poisoning attack is also threatening. It inserts backdoors in ML models by poisoning the training data. A common poisoning mechanism is to insert a secret trigger into the features and coax ML models to unknowingly learn the secret trigger as a pattern of the attacker's target label. The work of [17] demonstrates success poisoning attacks on lithography problems.

4.2 Attack Method on Model Performance

Adversarial attacks are based on the generation of adversarial samples as model inputs. The most fundamental attack method is fast gradient sign attack (FGSM) [7]. For attacks without a specific target, it perturbs the input features X towards the direction that maximizes the error between prediction $F(X)$ and the correct label y. To avoid the attack being perceived by victims, the perturbation is often constrained with a maximum perturbation amount.

For EDA applications, the attack scenario and algorithm can be quite different. For example, when targeting lithography hotspot detectors [19], instead of perturbing every pixel, the perturbation, in this case, can be inserting or deleting shapes of artificial sub-resolution assist features (SRAFs) on layouts. Also, the layout after the perturbation has to remain legal and DRC violation clean [19, 41]. Fig. 3 shows this special scenario for lithography tasks. Compared with the well-studied traditional pixel-level attack, this largely different attack constraint leads to different attack algorithms [19, 41] in EDA applications. The potential attackers are low-quality IP/macros providers who wish to hide lithography deficiencies in their design or maliciously sabotage the downstream manufacturing process.

In ML for EDA, adversarial attacks are threatening to ML models targeting lithography problems, where circuit layouts as inputs can easily come from malicious third-party providers. In comparison, for models supposed to be deeply incorporated and coupled

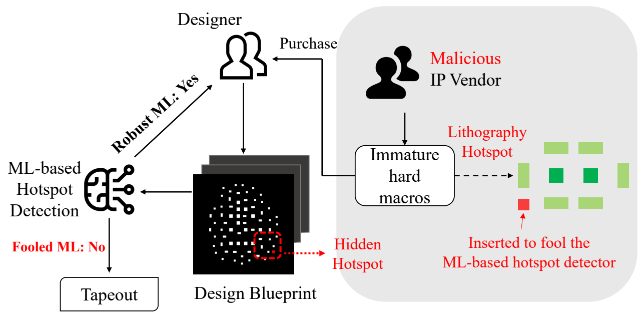


Figure 3: Malicious IP vendors may use adversarial perturbations to hide hotspots in immature designs. Designers using an unrobust ML-based hotspot detector for printability verification may suffer from a great loss at the tapeout stage due to hidden hotspots [26].

with existing design flows, in practice, it will be more difficult for attackers to insert their perturbations into model inputs.

In addition, although we introduce adversarial attacks by assuming attackers have access to white-box model F with known weights, actually they can also be applied to black-box scenarios. In this case, the adversarial samples can be generated based on certain surrogate models with similar functionality. These samples are still effective after transferring to the black-box target model F.

Besides adversarial attacks, poisoning attacks target the model training stage. Take the same lithography problem as an example, to hide lithography deficiencies, malicious insiders can stealthily introduce a backdoor into lithography detectors by providing poisoned training data with backdoor ‘triggers’. The detector is thus trained to link the trigger with a non-hotspot label. If this detector is adopted and deployed, any attackers knowing the backdoor can pass off a low-quality design as ‘hotspot-free’ by inserting the trigger in their own layouts [17]. Recent studies [18] show that this poisoning attack on lithography can be defended by diluting the intentional bias from triggers with data augmentation strategies.

4.3 Defense Method on Model Performance

To cope with potential adversarial attacks in ML for EDA, a most recent work [26] presents a robust ML-based lithography hotspot detector. It proposes an innovative regularization-based defense method named DRC-guided CURE. The CURE stands for a technique called curvature regularization [25]. It minimizes the curvature of the training loss to achieve robustness against adversarial attacks. To minimize the curvature of the loss function with respect to input sample X, the CURE regularizer should penalize large eigenvalues of the Hessian H of the loss function at the input sample, since the eigenvalues correspond to the amount of curvature at the direction of their corresponding eigenvectors. After approximation and simplification, the existing CURE regularizer is formulated as below [25]:

$$L_{\text{CURE}}(X) = ||\nabla\ell(X + hz) - \nabla\ell(X)||^2, \tag{1}$$

where the step z simply equals the sign of the gradient with normalization $z = \text{sign}(\nabla\ell(X))/||\text{sign}(\nabla\ell(X))||$, and the h controls the scale of this step. The rationale behind the choice of this step z is that it targets robustness against ℓ_∞-constrained perturbations. However, the ℓ_∞ constraint does not hold in adversarial perturbations to lithography hotspot layouts. This CURE method needs customization for the lithography attack scenario.

	Vanilla	AT	CURE	Guided CURE
Original Accuracy	0.743	0.690	0.820	**0.850**
After-Attack Accuracy	0.438	0.595	0.630	**0.690**
Attack Success Rate	0.408	0.440	0.231	**0.188**

Table 3: Comparison of hotspot accuracy before and after adversarial attacks when different defense methods are applied. Here, AT denotes adversarial training [26].

To improve the existing CURE method, this work [26] further analyzes and utilizes the regions that are potentially vulnerable to adversarial perturbation attacks. Potential perturbations include the insertion of new fraudulent SRAFs and the removal of preexisting SRAFs. The insertion of new SRAFs should remain DRC-clean and keep a minimum distance from existing polygons. These constraints enable the calculation of the potential adversarial attack regions.

Based on the calculation of attack regions, this work re-designs the CURE-based regulation term and proposes the DRC-guided CURE. The step function z in the aforementioned CURE function can be rewritten as a DRC-guided step $z' = (m_i - m_r)/||m_i - m_r||$, where m_i is the binary mask of the region for potential SRAF insertion (positive sign in z') and m_r is the binary mask of the region for potential SRAF removal (negative sign in z'). It replaces the step function z in Equation 1 with this new function z'. In this way, the defense algorithm is customized to this special type of attack in lithography.

4.4 Experiment of Defense Method on Model Performance

The effectiveness of this new DRC-guided-CURE method [26] is compared with both existing vanilla models and other baseline defense algorithms. The baseline defense algorithm first includes the plain CURE method. Another baseline defense algorithm is adversarial training (AT), which directly trains the ML model with adversarial-attacked samples. All model architectures are controlled to be the same as prior work [40].

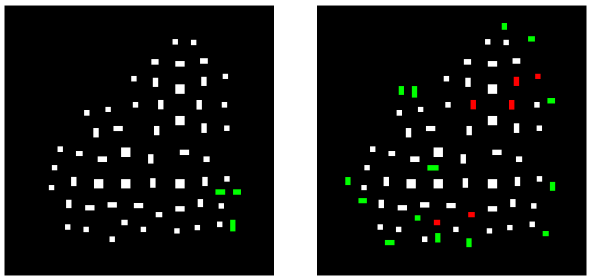
Table 3 [26] compares the performance of the proposed DRC-guided-CURE method and three other baselines. It evaluates 1) average accuracy before attack, 2) average accuracy after attack, 3) the attack success rate. The accuracy in Table 3 actually means the recall or true positive rate value. The guided-CURE training method achieves better accuracy both with and without adversarial attacks. The better accuracy is also verified by the lower attack success rate in Table 3.

Fig. 4 further shows the visualization of two adversarial attack examples on both the vanilla model and DRC-guided robust model [26]. Their difference is obvious. To fool the vanilla model, it only requires three fraudulent SRAF insertions in Fig. 4(a) and one preexisting SRAF deletion in Fig. 4(c). In comparison, the model trained with DRC-guided-CURE method is much more robust. It takes dozens of SRAF insertions or deletions to fool such a robust model in Fig. 4(b) and Fig. 4(d). It shows the higher robustness of this solution.

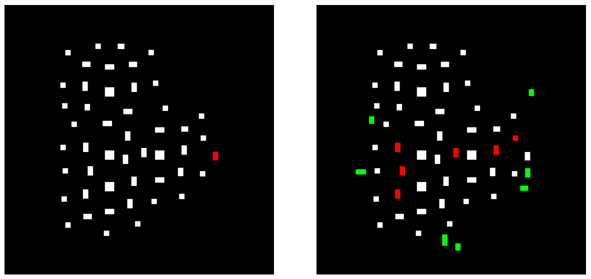
5 UNRELIABILITY IN ML PERFORMANCE

5.1 Model Unreliability Overview

We have discussed security concerns in ML for EDA caused by malicious attacks. In addition, the model reliability concern is also



(a) Fooling the vanilla model. (b) Fooling the robust model.



(c) Fooling the vanilla model. (d) Fooling the robust model.

Figure 4: Two examples of adversarial perturbations required to fool the ML-based lithography hotspot detector before and after DRC-guided CURE defense. (a)(b) Example one. (c)(d) Example two. Here, green blocks denote inserted fraudulent SRAFs, and red blocks denote deleted preexisting SRAFs [26].

worth attention. This challenge is reflected by unexpected model accuracy degradation on new test samples.

In practice, model developers often overestimate their models. It is a fallacy to believe ML models will always provide the same level of accuracy demonstrated in validation when it is eventually deployed "in the wild" for real applications. In many ML for EDA works, the held-out validation/test dataset for accuracy evaluation may be an insufficient representation of real model inputs. Such validation data is often too similar to the training data, since they are usually collected in a similar way, from similar designs, and using the same technology node and design flow. But currently, there are no techniques that support developers/users to know for which type of new test circuits, their ML model is no longer applicable. To the best of our knowledge, there is no systematic study on this topic. As a result, users cannot safely trust any ML model in EDA. It affects all scenarios mentioned in Table 1 and may become a major obstacle that prevents a wide application of ML in EDA in the future. Although this challenge is not caused by any malicious attackers, but can be serious due to two properties.

First, huge heterogeneity may exist between training and testing data samples, because of the large difference among circuits due to functionality, micro-architecture, and technology node. Some testing samples can be largely different from existing training samples, inevitably leading to accuracy degradation in these samples.

Second, it is very difficult for users to detect unexpected accuracy degradation on new testing samples. It requires ground-truth labels to be collected, which is highly time-consuming and inherently against the purpose of adopting ML models. Also, undetected accuracy degradation can lead to much less optimized design solutions or even chip failure, causing serious income loss for users from design companies.

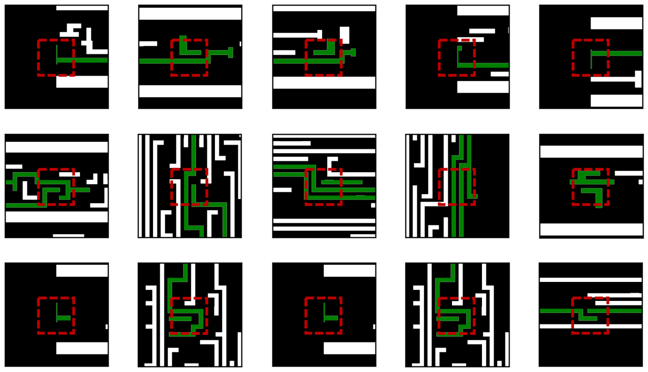


Figure 5: Generated model focus elements (in green) for lithography hotspot detection. All displayed samples have a hotspot (label is positive). Ground-truth hotspot locations are all centered in a red square, and this is confidential.

5.2 Solution from Model Interpretability

This paper proposes to address this model reliability challenge by improving the interpretability of existing ML solutions. It means explaining each ML prediction by indicating where the ML model is focusing in each input sample. Such *focus region* denotes the key part of an input sample that has an obviously larger impact on the ML prediction.

For an input sample $X \in \{0,1\}^{w \times h}$, we can extract a local region $x \in X$ from it, with the remaining parts denoted as $X - x$. After the extraction, we can generate model predictions on the selected part as $F(x)$ and the remaining part as $F(X - x)$. The target is to extract the *focus region* $x^* \in X$ that has the maximum impact on the ML prediction. We propose two metrics to evaluate such impact. Assume X is a positive test sample with $F(X) > 0$. First, when the part x is presented to the ML model alone, the prediction $F(x)$ should remain a large positive number. Second, after removing x, the prediction on the remaining part $X - x$ should drop significantly, perhaps to a negative number. It means the target focus region x^* should maximize $F(x)$ and minimize $F(X - x)$. In addition, we try to minimize the size of the extracted x, denoted as $S(x)$. The target focus region x^* is defined as $x^* = \text{argmax}_{x \in X} \frac{F(x)-F(X-x)}{S(x)}$. This target function can be viewed as the impact per unit area.

5.3 Experiment on Model Interpretability

Fig. 5 shows our preliminary results on the generated model focus elements x^* (in green) of an accurate ML model for the lithography hotspot detection task. According to the definition, the model predicts these samples as positive mainly because of these model focus elements. From the ML model's perspective, green elements are major contributors to hotspots. In comparison, the ground-truth hotspot locations are all centered in a red square, and this information is confidential to the model. As Fig. 5 shows, the generated model focus elements (in green) and real hotspot regions (in red box) overlap surprisingly well. We thus know the model is focusing on correct patterns.

With such an improvement in model interpretability, experienced users can inspect a prediction before trusting it. For example, users can check whether the model is making predictions based on correct input patterns. It thus helps to avoid unexpected wrong predictions on new test samples. By identifying major contributors (i.e. focus region) in input samples, this method can also enable the detection of backdoor triggers.

6 CONCLUSION

In this paper, we present the latest research advances regarding security and reliability concerns in ML for EDA tasks. They include an effective attack method for model extraction, a defense method for the model-performance attack, and a solution to the model unreliability problem. In the future, we will continue to explore more customized attack and defense methods with more in-depth experiments in ML for EDA.

ACKNOWLEDGEMENT

This research was supported by ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR. We thank Chen-Chia Chang and Jingyu Pan from Duke University for their thoughts and advice.

REFERENCES

[1] Erick Carvajal Barboza, Nishchal Shukla, Yiran Chen, and Jiang Hu. 2019. Machine learning-based pre-routing timing prediction with reduced pessimism. In *DAC*.
[2] Cadence. 2021. Cadence Cerebrus Intelligent Chip Explorer. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/cerebrus-intelligent-chip-explorer.html
[3] Chen-Chia Chang, Jingyu Pan, Zhiyao Xie, Jiang Hu, and Yiran Chen. 2023. Rethink before Releasing your Model: ML Model Extraction Attack in EDA. In *ASP-DAC*.
[4] Chen-Chia Chang, Jingyu Pan, Tunhou Zhang, Zhiyao Xie, Jiang Hu, Weiyi Qi, Chunwei Lin, Rongjian Liang, Joydeep Mitra, Elias Fallon, and Yiran Chen. 2021. Automatic Routability Predictor Development Using Neural Architecture Search. In *ICCAD*.
[5] Jingsong Chen, Jian Kuang, Guowei Zhao, Dennis J-H Huang, and Evangeline FY Young. 2020. PROS: A Plug-in for Routability Optimization applied in the State-of-the-art commercial EDA tool using deep learning. In *ICCAD*.
[6] Steve Dai, Yuan Zhou, Hang Zhang, Ecenur Ustun, Evangeline FY Young, and Zhiru Zhang. 2018. Fast and accurate estimation of quality of results in high-level synthesis with machine learning. In *FCCM*.
[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).
[8] Kourosh Hakhamaneshi, Nick Werblun, Pieter Abbeel, and Vladimir Stojanović. 2019. BagNet: Berkeley analog generator with layout optimizer boosted with deep neural networks. In *ICCAD*.
[9] Abdelrahman Hosny, Soheil Hashemi, Mohamed Shalan, and Sherief Reda. 2020. Drills: Deep reinforcement learning for logic synthesis. In *ASP-DAC*.
[10] Guyue Huang, Jingbo Hu, Yifan He, Jialong Liu, Mingyuan Ma, Zhaoyang Shen, Juejian Wu, Yuanfan Xu, Hengrui Zhang, Kai Zhong, et al. 2021. Machine learning for electronic design automation: A survey. *arXiv preprint arXiv:2102.03357* (2021).
[11] Yu-Hung Huang, Zhiyao Xie, Guan-Qi Fang, Tao-Chun Yu, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2019. Routability-driven macro placement with embedded CNN-based prediction model. In *DATE*.
[12] IBS. 2020. As Chip Design Costs Skyrocket, 3nm Process Node Is in Jeopardy. https://www.extremetech.com/computing/272096-3nm-process-node.
[13] Yoav Katz, Michal Rimon, Avi Ziv, and Gai Shaked. 2011. Learning microarchitectural behaviors to improve stimuli generation quality. In *DAC*.
[14] Hao Li, Fanshu Jiao, and Alex Doboli. 2016. Analog circuit topological feature extraction with unsupervised learning of new sub-structures. In *DATE*.
[15] Rongjian Liang, Zhiyao Xie, Jinwook Jung, Vishnavi Chauha, Yiran Chen, Jiang Hu, Hua Xiang, and Gi-Joon Nam. 2020. Routing-free crosstalk prediction. In *ICCAD*.
[16] Hung-Yi Liu and Luca P Carloni. 2013. On learning-based methods for design-space exploration with high-level synthesis. In *DAC*.
[17] Kang Liu, Benjamin Tan, Ramesh Karri, and Siddharth Garg. 2020. Poisoning the (data) well in ML-based CAD: A case study of hiding lithographic hotspots. In *DATE*.
[18] Kang Liu, Benjamin Tan, Gaurav Rajavendra Reddy, Siddharth Garg, Yiorgos Makris, and Ramesh Karri. 2020. Bias Busters: Robustifying DL-based Lithographic Hotspot Detectors Against Backdooring Attacks. *TCAD* (2020).
[19] Kang Liu, Haoyu Yang, Yuzhe Ma, Benjamin Tan, Bei Yu, Evangeline FY Young, Ramesh Karri, and Siddharth Garg. 2020. Adversarial perturbation attacks on ML-based cad: A case study on CNN-based lithographic hotspot detection. *TODAES* (2020).
[20] Mingjie Liu, Keren Zhu, Jiaqi Gu, Linxiao Shen, Xiyuan Tang, Nan Sun, and David Z Pan. 2020. Towards decrypting the art of analog layout: Placement quality prediction via transfer learning. In *DATE*.
[21] Yi-Chen Lu, Jeehyun Lee, Anthony Agnesina, Kambiz Samadi, and Sung Kyu Lim. 2019. GAN-CTS: A generative adversarial framework for clock tree prediction and optimization. In *ICCAD*.
[22] Yi-Chen Lu, Sai Surya Kiran Pentapati, Lingjun Zhu, Kambiz Samadi, and Sung Kyu Lim. 2020. TP-GNN: A graph neural network framework for tier partitioning in monolithic 3D ICs. In *DAC*.
[23] Yuzhe Ma, Haoxing Ren, Brucek Khailany, Harbinder Sikka, Lijuan Luo, Karthikeyan Natarajan, and Bei Yu. 2019. High performance graph convolutional networks with applications in testability analysis. In *DAC*.
[24] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. 2021. A graph placement methodology for fast chip design. *Nature* (2021).
[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Jonathan Uesato, and Pascal Frossard. 2019. Robustness via curvature regularization, and vice versa. In *CVPR*.
[26] Jingyu Pan, Chen-Chia Chang, Zhiyao Xie, Jiang Hu, and Yiran Chen. 2022. Robustify ML-Based Lithography Hotspot Detectors. In *ICCAD*.
[27] Martin Rapp, Hussam Amrouch, Yibo Lin, Bei Yu, David Z Pan, Marilyn Wolf, and Jörg Henkel. 2021. MLCAD: A Survey of Research in Machine Learning for CAD Keynote Paper. *TCAD* (2021).
[28] Haoxing Ren, George F Kokai, Walker J Turner, and Ting-Sheng Ku. 2020. ParaGraph: Layout parasitics and device parameter prediction using graph neural networks. In *DAC*.
[29] Haralampos-G Stratigopoulos and Yiorgos Makris. 2008. Error moderation in low-cost machine-learning-based analog/RF testing. *TCAD* (2008).
[30] Synopsys. 2021. DSO.ai: AI-Driven Design Applications. https://www.synopsys.com/implementation-and-signoff/ml-ai-design/dso-ai.html
[31] Zhiyao Xie, Guan-Qi Fang, Yu-Hung Huang, Haoxing Ren, Yanqing Zhang, Brucek Khailany, Shao-Yun Fang, Jiang Hu, Yiran Chen, and Erick Carvajal Barboza. 2020. FIST: A feature-importance sampling and tree-based method for automatic design flow parameter tuning. In *ASP-DAC*.
[32] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Jiang Hu. 2018. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *ICCAD*.
[33] Zhiyao Xie, Shiyu Li, Mingyuan Ma, Chen-Chia Chang, Jingyu Pan, Yiran Chen, and Jiang Hu. 2022. DEEP: Developing Extremely Efficient Runtime On-Chip Power Meters. In *ICCAD*.
[34] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Chen-Chia Chang, Jingyu Pan, and Yiran Chen. 2022. Pre-Placement Net Length and Timing Estimation by Customized Graph Neural Network. *TCAD* (2022).
[35] Zhiyao Xie, Rongjian Liang, Xiaoqing Xu, Jiang Hu, Yixiao Duan, and Yiran Chen. 2021. Net^2: A Graph Attention Network Method Customized for Pre-Placement Net Length Estimation. In *ASP-DAC*.
[36] Zhiyao Xie, Jingyu Pan, Chen-Chia Chang, Jiang Hu, and Yiran Chen. 2022. The Dark Side: Security and Reliability Concerns in Machine Learning for EDA. *TCAD* (2022).
[37] Zhiyao Xie, Haoxing Ren, Brucek Khailany, Ye Sheng, Santosh Santosh, Jiang Hu, and Yiran Chen. 2020. PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network. In *ASP-DAC*.
[38] Zhiyao Xie, Xiaoqing Xu, Matt Walker, Joshua Knebel, Kumaraguru Palaniswamy, Nicolas Hebert, Jiang Hu, Huanrui Yang, Yiran Chen, and Shidhartha Das. 2021. APOLLO: An Automated Power Modeling Framework for Runtime Power Introspection in High-Volume Commercial Microprocessors. In *MICRO*.
[39] Biying Xu, Yibo Lin, Xiyuan Tang, Shaolan Li, Linxiao Shen, Nan Sun, and David Z Pan. 2019. Wellgan: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout. In *DAC*.
[40] Haoyu Yang, Jing Su, Yi Zou, Yuzhe Ma, Bei Yu, and Evangeline FY Young. 2018. Layout hotspot detection with feature tensor generation and deep biased learning. *TCAD* (2018).
[41] Haoyu Yang, Shifan Zhang, Kang Liu, Siting Liu, Benjamin Tan, Ramesh Karri, Siddharth Garg, Bei Yu, and Evangeline FY Young. 2021. Attacking a CNN-based Layout Hotspot Detector Using Group Gradient Method. In *ASP-DAC*.
[42] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. 2018. Developing synthesis flows without human knowledge. In *DAC*.
[43] Han Zhou, Wentian Jin, and Sheldon X-D Tan. 2020. GridNet: Fast data-driven EM-induced IR drop prediction and localized fixing for on-chip power grid networks. In *ICCAD*.
[44] Yuan Zhou, Haoxing Ren, Yanqing Zhang, Ben Keller, Brucek Khailany, and Zhiru Zhang. 2019. PRIMAL: Power Inference using Machine Learning. In *DAC*.