The Dark Side: Security and Reliability Concerns in Machine Learning for EDA

Zhiyao Xie, Jingyu Pan *Graduate Student Member, IEEE*, Chen-Chia Chang,
Jiang Hu *Fellow, IEEE*, Yiran Chen *Fellow, IEEE*

***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 unprecedented efficient EDA methods have been enabled by machine learning (ML) techniques. While ML demonstrates its great potential in circuit design, however, the dark side about potential security and model reliability problems, is seldomly discussed. This paper gives a comprehensive and impartial summary of all security and reliability concerns we have observed in ML for EDA. Many of them are hidden or neglected by practitioners in this field. In this paper, we first provide our taxonomy to define four major types of concerns, then we analyze different application scenarios and special properties in ML for EDA. After that, we present our detailed and impartial analysis of each type of concern with experiments.**

***Index Terms*—Machine learning, physical design, security, design privacy, model reliability.**

I. Introduction

Driven by the continuously growing complexity in 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 [1]. Therefore, there is a compelling need for essential improvement on IC design efficiency through new methodologies and design automation techniques. To solve this, machine learning (ML) techniques are considered a highly promising direction.

In recent years, machine learning for EDA has become a trending topic [2], [3]. 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 [2], [3]. Some techniques are further adopted in commercial EDA tools [4], [5]. 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.

Existing ML for EDA techniques seek various attractive properties, such as better design quality, shorter turn-around time, and a higher level of automation. A significant amount of research and engineering efforts have been invested in these targets. However, these properties are no longer desirable if fundamental *security* and *model reliability* requirements are not first satisfied. In this study, we focus on the seldomly discussed dark side by trying to cover all measures about causing and preventing unforeseen consequences in ML for EDA.

Actually, as ML is introduced in design automation, unprecedented security and model reliability concerns arise, but most practitioners are not fully aware of them. According to our study, the negligence of these potential problems can lead to serious consequences for both model providers and users. Possible consequences include misleading results, design information leakage, model information leakage, etc. While a few previous works [6], [7], [8], [9] studied possible adversarial attacks on ML models targeting lithography problems, they only account for a very small portion of potential challenges we observed in ML for EDA. In this paper, we try to give a more comprehensive and impartial study on all identified challenges in ML for EDA. We propose our taxonomy to first define three major types of security concerns, which are all caused by malicious attacks:

1) **Attacks against data privacy**, e.g., attacks that try to infer private information about design data. The victims in this case are the data providers who expect protections of their data. The attackers can be malicious competitors targeting access to private data by exploring their access to ML models.

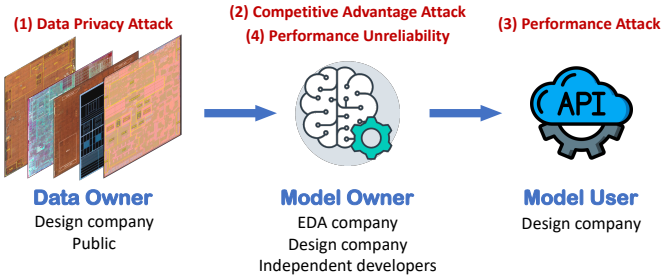


Fig. 1: Illustration of one typical ML for EDA flow.

Manuscript received March 20, 2022; revised June 22, 2022; accepted August 10, 2022. This work was supported in part by the National Science Foundation under Grant NSF-2106828, NSF-2106725, and Semiconductor Research Corporation (SRC) GRC-CADT 3103.001 and 3104.001. This article was recommended by Associate Editor S. Ghosh. *(Corresponding author: Zhiyao Xie.)*

Zhiyao Xie is with the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology, Hong Kong SAR (email: eezhiyao@ust.hk).

Jingyu Pan, Chen-Chia Chang, and Yiran Chen are with the Department of Electrical and Computer Engineering at Duke University, Durham, NC 27708, USA (email: jingyu.pan@duke.edu; chenchia.chang@duke.edu; yiran.chen@duke.edu).

Jiang Hu is with the Department of Electrical and Computer Engineering at Texas A&M University, College Station, TX 77843, USA (email: jianghu@tamu.edu).

2) **Attacks against competitive advantage**, e.g., attacks that construct similar substitute models, which impair the competitive advantage of the original model. The victims are the model providers who wish to make 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 accuracy degradation on specific testing samples. Victims are model users, and attackers can be someone who wish to fool the model and introduce design deficiencies.

In addition, malicious attacks are not the only source of concern in this paper. We also pay attention to model unreliability problems, which cause unforeseen consequences in many scenarios and are especially serious in EDA and chip design. It constitutes the fourth concern studied in this paper:

4) **Inherent unreliability in ML performance**, e.g., unexpected accuracy degradation on new testing samples. Victims are model users, and there are no attackers in this concern.

Figure 1 illustrates a typical ML for EDA development and usage flow, and corresponding concerns. According to our taxonomy, all aforementioned previous studies [6], [7], [8], [9] can be categorized into the third type. In addition, a recent survey [3] on ML for EDA mentioned their concern about type-one attack on training data without a more detailed analysis.

In this paper, we present a comprehensive study with our preliminary experimental results on all identified security and model reliability concerns. According to our observation, some of these concerns/challenges are actually less practical, while others pose a high threat to data owners, model owners, or users. We will first present representative works, application scenarios, and special properties of ML for EDA in Section II. After that, these four major concerns are presented in Section III, IV, V, and VI, respectively. Finally, in Section VII, we discuss other potential concerns and impacts in the future, when ML for EDA becomes ubiquitous.

II. ML for EDA Background

A. Existing ML Solutions in EDA

We start with a brief inspection of representative 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 [10], [11] or design space exploration [12], [13]. Many power models [14], [15] are also proposed in early design stages. Some power models [16], [17] are further implemented for runtime circuit management. At logic synthesis, ML models are proposed for chip quality prediction [18], [19] and optimization [20], [21]. During physical design, more models perform predictions or optimizations on almost all important design metrics, including timing [22], [23], [24], macro placement [25], [26], routability [27], [28], [29], IR drop [30], [31], [32], clock tree quality [33], interconnect [34], crosstalk [35], 3D integration [36], etc. Also, ML models are developed for design verification [37], [38], design for testability (DFT) [39], and lithography problems [40], [41]. Besides the methods applied at specific design stages, automatic design flow tuning is another well explored topic in ML for EDA [42], [43].

ML-based methods are of course not only limited to digital designs. For analog design, similarly, various models have been developed for topology design [44], [45], device sizing [46], [47], pre-layout prediction [48], [49], layout evaluation [50], [51], layout generation [52], [53], and analog design testing [54]. For a more complete survey on all existing research efforts, please refer to previous survey papers [2], [3] solely devoted to this topic.

Besides being a hot research topic in academia, ML-based estimators have also gained popularity in the EDA industry. Recent versions of commercial tools already support the construction of ML models on delay [55] or congestion predictions [56], providing improved PPA or faster convergence after invoking the ML models in their tools [55], [56]. In addition, EDA vendors have provided ML models for design space exploration or design flow tuning, named DSO.ai [4] and Cerebrus [5].

Among these ML applications targeting digital or analog designs, almost all popular ML techniques have been applied. Most methods in ML for EDA adopt supervised models, especially neural network techniques, while some others perform reinforcement learning. In this work, we also focus on the most popular supervised methods. Considering the popularity in both EDA academia and industry, we believe ML models will play a more important role in design automation in the future. Therefore, a deep understanding of all potential security and model reliability concerns is essential.

B. Application Scenarios

To better analyze all security and model reliability concerns in ML for EDA, we should first fully understand the practical applications scenarios of these ML-based techniques. However, as an emerging type of chip design technique, new explorations in ML for EDA solutions are still ongoing while the pace of commercialization in the industry lags behind. Thus, besides observing existing solutions, we have to anticipate possible application scenarios in the near future.

Currently, many existing research efforts in ML for EDA merely target the demonstration of their correctness and effectiveness. A small portion of works have been verified and applied in private in-house design flows in design companies. In addition, some ML models are deployed in EDA tools by EDA vendors. They correspond to two major types of application scenarios. 1) Same model providers and users. For ML models developed and deployed for in-house flows internally, the model provider and user are from the same company and work rather closely. 2) Separate model providers and users. As Figure 1 shows, there may be separate model providers from EDA vendors or independent developers and model users from design companies. In the future, we tend to believe it is more likely for more ML model providers and users to be separated, like the separation of IC design, EDA, and fabrication in the semiconductor industry history.

TABLE I: Possible Application Scenarios of ML for EDA

Scenario	Black-Box	Trained	Separated Provider & User
S1	✓	✓	✓
S2	✗	✓	✓
S3	✓	✗	✓
S4	✗	✓	✗

Despite these observations, the anticipation of future application scenarios is not straightforward. Compared with traditional EDA software, ML for EDA methods adopt a different and more complex flow, which consists of multiple stages, including model architecture design, data and label collection, model training, model inference/prediction, and utilization of prediction results. These tasks could be divided differently between model providers and clients. Different partitions of tasks lead to different scenarios.

Table I presents four possible 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 [57]. Such cloud services allow model providers to charge users for queries. These ML models are of high commercial values. In this case, models will be vulnerable to attacks against competitive advantage, attacks against ML performance, and also unreliability problems.

In addition, there could be a special case, S2, where ML models are actually white-box to users or potential attackers. There are a few possible reasons causing the model to be white-box. For example, researchers, individual developers, and even companies may hope to directly open-source their trained model for free. Also, models targeting black-box in S1 may be hacked, especially if they are deployed locally instead of through cloud platforms without enough security measures. In this scenario, the ML model itself is already available to potential attackers, while new security concerns about the training design data privacy arise.

Another possible scenario, S3, is to leave more tasks to users. The model providers only design their ML methodology without performing the training. The method is provided as black-box, with information like feature, architecture, and optimization procedure not explicitly disclosed. Then users can train and use their own customized ML models as black-box with their own labeled data. Rather than being provided as stand-alone 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 [55] from vendors.

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 S4 in Table I. In this case, this rather private flow will be much less vulnerable to malicious attacks. But it will still be affected by the inherent unreliability of ML models, which will be covered in detail in Section VI.

C. Overview of Special Properties

Before giving a detailed analysis of all four types of challenges, we briefly inspect some special properties of ML for EDA solutions. Although many ML for EDA solutions have been developed based on black-box use of existing ideas from the ML community, we still observe some remarkable properties different from general ML tasks.

Unprecedented data heterogeneity: Huge heterogeneity can exist between data samples, resulting from the large difference among circuit designs due to functionality, micro-architecture, and technology node. For example, assuming we already restrict the training and testing data of an ML model to be only from Arm processors, we still cannot expect the model trained on old designs like Cortex-M0 with 40nm technology node to perform very well on latest designs like Neoverse N2 with 5nm technology. This level of training and testing data heterogeneity is uncommon in benchmarks for general ML applications like computer vision.

High complexity in data and pattern: A circuit contains orders-of-magnitude more information than an ordinary image. For prediction tasks, models are learning behaviors of highly complex EDA engines. For optimization tasks like macro placement, models are exploring a huge solution space [25], significantly larger than the Go game solved by AlphaGo [58]. These complexities increase the difficulty in studying security and reliability problems in ML for EDA.

More confidential design in higher demand: The construction of ML models in EDA relies on training data generated from circuit designs, which are highly confidential to design companies. Due to the aforementioned data heterogeneity, for ML models targeting applications on most cutting-edge circuit designs, similarly latest cutting-edge circuits are typically desired as training data for model construction. This tends to put these advanced highly confidential circuit designs at a higher risk of information leakage.

Potentially decentralized training data: Many ML for EDA developers have very limited access to the latest design data owned by design companies. Therefore, training with decentralized private circuit data is explored in recent works [59]. They propose to perform collaborative training on decentralized data with techniques like federated learning [60]. Such a scenario can lead to many additional risks.

Models performing binary classification or regression: Most security studies in general ML tasks target common multi-class classifiers. For example, there are 1000 classes in Image-Net benchmark for convolutional neural network (CNN) models and 3 classes in COLLAB benchmark [61] for graph neural network (GNN) models. In comparison, most predictive models in EDA perform binary classification or regression, while optimization models adopt reinforcement learning. This difference makes many attack and defense methods targeting multi-class classifiers no longer applicable. For example, some attack methods [62] utilize models' multi-class predictions to evaluate the model's 'confidence level' on any specific target class. Such 'confidence level' can leak training data properties.

D. Overall Experiment Setup

In this paper, we try to cover all security or reliability concerns we have observed in ML for EDA in these years. Since not all concerns have been systematically studied in ML for EDA before, we perform some preliminary experiments ourselves to better demonstrate our ideas. The experiments are mainly performed on two most representative and well-studied topics in ML for EDA, which are the routability problem during layout, and the lithography problem for manufacturing. The setup and dataset of these experiments mainly follow the most recent works [63], [8] on these topics.

For routability tasks, either routing congestions [29], [28] or DRC (design rule checking) [27], [64] are adopted as the metric of routability. The congestion detection is simpler than DRC violation detection in practice. Therefore, congestion detection models generally achieve higher accuracy.

Most experiments in this work are based on a comprehensive dataset using 74 designs with largely varying sizes from multiple benchmarks. There are 29 designs from ISCAS'89 [65], 13 designs from ITC'99 [66], 19 other designs from Faraday and OpenCores in the IWLS'05 [67], 13 designs from ISPD'15 [68]. For each design, multiple placement solutions are generated with different logic synthesis and physical design settings. Altogether 7,131 placement solutions are generated from these 74 designs. We apply Design Compiler® for logic synthesis and Innovus® [55] for physical design, with the NanGate 45nm technology library [69].

Besides routability tasks, we also conduct experiments on lithography hotspot detection, another representative topic in ML for EDA, to study relevant security concerns on adversarial attacks. The lithography hotspot detectors are also CNN-based. The experiment is based on a lithography dataset from the previous work [8], with four groups of 400 hotspot clips for adversarial sample generation and 34356 layout clips for model training.

We adopt a general set of notations that describe both routability and lithography problems. Some commonly used notations are summarized in Table II. Given a differentiable ML model F with trained weights W, denote the input features and the label of a training sample as X and y, respectively. For layout with width d and height h, the corresponding input features X usually include multiple two-dimensional features, describing the distribution of macros or blockages. Each feature is in $\mathbb{R}^{d\times h}$ and $X \in \mathbb{R}^{d\times h\times C}$, where C is number of features. The label y is either a two-dimensional distribution indicating locations of actual violations ($\mathbb{R}^{d\times h}$) or a scalar $\mathbb{R}$ indicating the overall quality of the layout.

$$F(X|W) : X \in \mathbb{R}^{d\times h\times C} \rightarrow y \in \mathbb{R}^{d\times h} \text{ or } y \in \mathbb{R}$$

The prediction from trained model F with weights W on each sample (X, y) is $F(X|W)$, which is also denoted as p. The shape of the prediction p is the same as y.

For routability prediction on both DRC violation and congestions, features X can include wire density distributions, blockage locations describing pins, cells, macros, and detailed pin shapes. For lithography hotspot predictions, features X include blockage locations describing vias and sub-resolution assistant features (SRAFs). A more detailed introduction and visualizations of features X and of actual labels y are provided in related prior works [27], [29], [8], [6].

X	Input features	y	Label
d	Layout width	h	Layout height
F	ML model	W	Model weights
p	Model predictions	μ, σ	Batch statistics
L	Error term in loss	R	Regularizer
X_r	Recovered features in the first attack		
F_a	Attack model in the second attack		
X_u	Unlabeled data in the second attack		

TABLE II: Commonly used notations in this manuscript.

III. Attacks Against Design Privacy

A. Design Privacy Overview

Training data is the foundation of ML for EDA and it directly determines the quality of ML models. Such data includes both input features and ground-truth labels. For a circuit design/IP used for data generation, input features are different representations of the design, and labels are corresponding circuit qualities including power, performance, etc. A circuit is significantly more complex than an ordinary image, thus provides rich information for model development. Such information can be highly confidential for design companies.

In the semiconductor industry, knowing competitors' development decisions easily provides a great competitive advantage. Previous studies [70], [71] have demonstrated that given an ML model, it is possible for attackers to reconstruct or recover sensitive feature information in the model training data. The process of maliciously recovering input features is commonly referred to as *model inversion* or *reconstruction attack* [71]. In ML for EDA, such attacks may cause serious security challenges on circuit designs/IPs used in training. Even compared with other ML applications involving private data, like medical image processing or language models on smartphones, attacks targeting ML for EDA models are more threatening, since attackers do not require high-quality recovery of training data. A very small part of information about the circuit design may already benefit the attacker. For example, attackers may only target basic information like dynamic scaling granularity, target manufacturing process, flat/horizontal implementation methodologies, etc. Based on the small piece of reconstructed features, it is possible for attackers with sufficient background to infer valuable information about the research or development direction of their target company.

To make things worse, as mentioned in Subsection II-C, in ML for EDA, due to data heterogeneity, more confidential design is in higher demand as high-quality training data. This property tends to put those most advanced and confidential circuit designs at a high risk of information leakage. This concern on design privacy is recognized as an open challenge by the recent ML for EDA survey [3].

B. Attack Method on Design Privacy

We provide a demonstration of the malicious reconstruction of training data in ML for EDA. It applies to most complex

ML models like deep neural networks. However, it turns out that such an attack has very high requirements on information available to attackers.

The fundamental attacking mechanism is straightforward. Based on the setting presented in Subsection II-D and summarized in Table II, attackers can try to reconstruct similar input features of the sample, denoted as X_r, targeting $X_r \approx X$. This X_r can be referred to as *reconstructed input*. The ultimate target can be formalized as below.

$$\text{Based on model } F, \text{ find } X_r = argmin_{X_r}||X_r - X||_2 \quad (1)$$

When the attacker can access the model as white-box, as indicated by scenario S2 in Table I, he has full knowledge of the weights W. However, the information about the ML model itself is not enough. We apply a very strong assumption to study the most threatening case of such an attack. If an attacker targets the training sample, we assume he/she can generate or hack a close estimation of the model prediction $p' \approx p = F(X|W)$ of this sample. This assumption is also made in representative reconstruction attack works [71] on facial image models. The attack method starts with an initial generation of the X_r with random signals. After that, gradient descent with respect to X_r is performed iteratively, as shown below, until it reaches convergence.

$$\text{In each iteration, } X_r \; -= \; \nabla_{X_r} Loss(F(X_r|W),\; p') \quad (2)$$
$$= \; \nabla_{X_r}||F(X_r|W) - p'||_2 \quad (3)$$

Different from the model training process, where gradient descent is performed with respect to model weights w, in this attack, gradient descent is performed with respect to the reconstructed input X_r. This operation minimizes the difference between the prediction $F(X_r|W)$ based on attacker-reconstructed input X_r and the actual prediction p based on X. By performing this, X_r is optimized to approximate the original training data sample X.

However, in practice this simple loss function $||F(X_r|W) - p'||_2$ does not work well. Simply minimizing the difference between original and new model output may not optimize the reconstructed input X_r towards the original feature X. This is also verified in our own experiment. Instead, extra loss function terms have to be introduced to steer the optimization direction and enforce the similarities between X_r and original feature X [72].

To improve the attack quality, we provide additional guidance to make the X_r follow existing feature statistics, which are stored in widely-used batch normalization (BN) layers of deep neural networks. This is inspired by the work of [72] in computer vision. The BN layer [73] normalizes the feature maps during training and implicitly captures the channel-wise running/moving means μ_{BN} and variances σ^2_{BN}. Therefore, we can steer the mean μ and variance σ^2 of input batches with reconstructed input X_a towards the running values stored in all BN layers. We define regularization terms for the l^{th} BN layer with μ_{BN_l} and $\sigma^2_{BN\_l}$, as shown below.

$$R^l(X_r) = ||\mu_l(X_r) - \mu_{BN\_l}||_2 + ||\sigma^2_l(X_r) - \sigma^2_{BN\_l}||_2$$

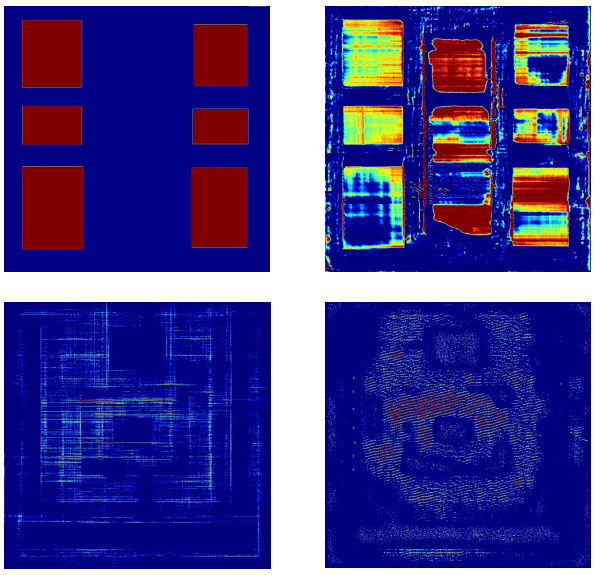
where $\mu_l(X_r)$ and $\sigma^2_l(X_r)$ are the mean and variance of the batch with reconstructed input X_r at the l^{th} BN layer. Then these penalty terms corresponding to all BN layers are added to the loss function, with a controllable weight α.

$$Loss(F(X_r|W), p') = ||F(X_r|W) - p'||_2 + \alpha \sum_l R^l(X_r) \quad (4)$$

In this way, the extra regularization steers the optimization of X_r towards the recovery of original training features X.

C. Experiment on Design Privacy Attack

Experimental results on model inversion attack with loss in Equation 4 are shown in Figure 2. We present our inversion results on two routability prediction features: macro positions and density of all net bounding boxes. The original features are shown in Figure 2(a) and the features reconstructed by attackers are in Figure 2(b). A certain degree of similarities can indeed be observed, especially in large-scale patterns. This example in Figure 2 is representative in our experiment and the rest of the dataset shows a similar trend. We first provide our qualitative observation on it. For the macro locations, the sizes and locations all six macros in Figure 2(a) are reconstructed by X_r in Figure 2(b). For net bounding boxes, similarly, the regions with high net density are reconstructed. However, obvious differences still exist in both large-scale and small-scale patterns. For macros in Figure 2(b), three false-positive macros are generated in the middle. Similarly, there are also false-positive net bounding box densities reconstructed in originally empty regions. Besides observations, we further provide a quantitative measurement of the similarities. The similarity is measured with ROC curve AUC like a prediction



(a) (b)

Fig. 2: Malicious model inversion attack. (a) The feature X in training data (left column). (b) The reconstructed input X_r (right column). Target features on: the distribution of macros (1^{st} row) and net bounding boxes (2^{nd} row).

problem, with original features X viewed as the label and reconstructed X_r viewed as its prediction. The AUC ranges from 0 to 1, with AUC = 0.5 indicating the accuracy of random guessing. For recovered macros and net bounding boxes, their similarity measured with ROC AUC is 0.77 and 0.67, respectively. It indicates limited similarity between X and X_r, and is consistent with the aforementioned observation.

More importantly, we emphasize that such attack is already based on a few very strong assumptions: (1) Attacker has white-box access to the ML model; (2) Attacker has an approximation of the prediction value p'. While the first condition may be achieved by hacking in scenario S2 or building very similar surrogate models, the generation of p' is very difficult in practice. Despite these strong assumptions and the carefully designed attack algorithm in Equation 4, we still get limited performance on design privacy attack, as reflected in Figure 2. Therefore, we conclude that based on existing techniques and our current exploration, the overall difficulty to conduct a model inversion attack on design privacy in ML for EDA is actually high.

In order to defend against such an attack, direct white-box sharing of trained models to the public or untrusted third parties should first be avoided. This rule will be enforced at the model sharing stage after model development. Also, as indicated in Equation 2, the attack relies on loss gradients. Some works [74] propose the idea of limiting gradient values below a certain threshold during training, in order to defend against input reconstruction attacks. This will be applied during the model development stage.

IV. Attacks Against ML Model Competitiveness

A. ML Models Competitiveness Overview

As indicated by scenario S1 in Table I, 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 these high-quality models, with steps including data collection, label generation, ML model design, ML model training and validation, etc. To provide even better service, they may have to construct multiple ML models for different types of design and technologies, taking extra engineering efforts. In summary, these trained ML models are important business assets and are costly to develop.

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 much lower cost. This malicious attack is 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.

In addition, aforementioned scenario S1 in Table I is not the only vulnerable business model. In scenario S3, where only ML model architecture is provided as black-box without performing the training, malicious attacks are also possible. Attackers may infer the model architecture, in order to save their own research cost. In general ML applications, this has been achieved by building an extra ML model to map from the concatenation of query outputs to the model architecture attributes [75]. It can be further improved by crafting own training data that maximizes information leakage. However, this attack on model architecture has only been verified on very simple models with less than 5 convolutional layers [75].

B. 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 existing trained ML models. The most fundamental yet effective attack methodology is to generate pseudo labels by querying the MLaaS-provided model F with attackers' own unlabeled data. In practice, label generation is one of the most costly steps during model development in ML for EDA. First, it can take a large computation cost and long runtime to finish a design flow and get accurate simulation results, which are the labels. For example, assuming we work on a design with more than one million gates, it easily takes more than one day to finish synthesis and physical design to generate one complete layout. If developers plan to generate 1,000 labeled samples on designs at this level of complexity, it will take dozens of machines running for months. Second, this label generation process requires licenses of commercial tools. Third, it requires great engineering expertise and efforts to generate reasonable and realistic training labels. In summary, label generation requires extensive computation resources, commercial EDA tool licenses, engineer efforts, time, etc.

If potential competitors/attackers can skip the label generation process to build their own dataset, the model construction will be much easier. We refer the provided MLaaS black-box model as the *victim model* F with trained weights W and the ML model developed by attackers as the *attack model* F_a with weights W_a. Given unlabeled input data X_u, the attacker can query the victim model $F(X_u|W)$ and use it as the pseudo label to train the attack model F_a. So the attack method is a very simple gradient descent optimization. In each iteration,

$$W_a \ -= \ \nabla_{W_a} Loss(F_a(X_u|W_a),\ F(X_u|W))$$

This is the most fundamental while effective attack targeting scenario S1. Based on this, attackers may further reduce the number of queries, in order to save the cost. For example, they can choose to select and only query the most representative unlabeled samples, based on ideas from active learning or semi-supervised learning.

C. Experiment on Model Competitiveness Attack

We demonstrate the effectiveness of our proposed fundamental model extraction attack in the routability experiment on constructing congestion models. Following the aforementioned scenario, we divide all of our existing data into four partitions without any overlap: 1) 40% of labeled data used to train the original victim model. 2) 10% labeled data used for

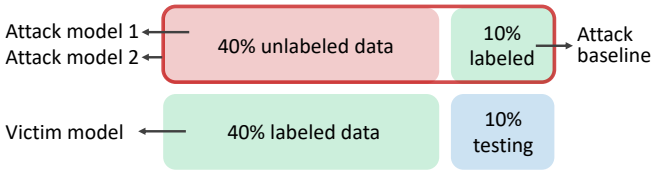


Fig. 3: The partitioning of dataset to study model competitiveness attack. Performance of these models are in Table III.

testing model accuracy. 3) 40% of unlabeled data prepared by attackers. 4) 10% labeled data prepared by attackers, in order to build a baseline. Figure 3 illustrates the data partitioning in this setup. Notice that the 40% unlabeled data from attackers are different from the 40% labeled training data of the victim model. This is very close to a realistic scenario, where attackers use different data from model developers.

Based on the data partition, an attack baseline is first trained with 10% labeled data. Then an attack model 1 is trained on 40% unlabeled data with pseudo labels from victim model F. No actual label is provided by the attacker for this model. An attack model 2 is trained on both 40% unlabeled data and the 10% labeled data.

Table III shows performance comparisons between the original victim model F, attack model baseline, and two attack models F_a. For attack model 1, without any labeled data, it achieves an accuracy of AUC=0.796, which is close to the victim model. For attack model 2, with a small portion (10%) of extra labeled data, it achieves even higher accuracy (AUC=0.811) than the victim model. These results demonstrate the effectiveness of model extraction attack with such a simple pseudo-labeling method.

According to the result in Table III, attackers can train even more accurate models with a very small portion of labeled data by querying the victim model. This attack proves to be efficient and profitable. It poses a threat to the competitiveness and business value of provided models in ML for EDA.

As for possible countermeasures for this attack, it is hard to prevent such malicious model extraction directly [76]. To make the attack more difficult, the model should return final predictions e.g. 0 / 1 / $\{0,1\}^{d\times h}$, instead of raw model output with confidence information, e.g. 0.4 / 0.6 / $\mathbb{R}^{d\times h}$. This only requires simple adjustments when returning query results. Some works [77] further propose to detect malicious model-extraction queries assuming they try to explore decision boundaries, which will result in a different distribution compared with normal queries. This mechanism requires an extra detection process when model processing queries.

V. Attacks Against ML Performance

A. ML Performance Attack Overview

Besides aforementioned attacks targeting data privacy or model competitiveness, another main type of malicious attacks may happen in ML for EDA targets affecting the performance of existing ML models. Compared with the previous two types of attacks, which are less explored by ML for EDA community, some prior works [6], [7], [8] studied the attack on the performance of CNN-based lithography hotspot detectors.

Model	Training data	Accuracy (AUC)
MLaaS-provided victim	40% labeled data	0.806
Attack baseline	10% labeled data	0.765
Attack model 1	40% unlabeled data	**0.796**
Attack model 2	40% unlabeled data + 10% labeled data	**0.811**

TABLE III: Attack on model competitiveness. The MLaaS-provided (victim) model and attackers use different data.

There exist 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. Such adversarial perturbation makes use of the inherent susceptibility of deep neural networks. However, in practice, it may not be feasible for outside attackers to easily modify the input in an ML-integrated circuit design flow.

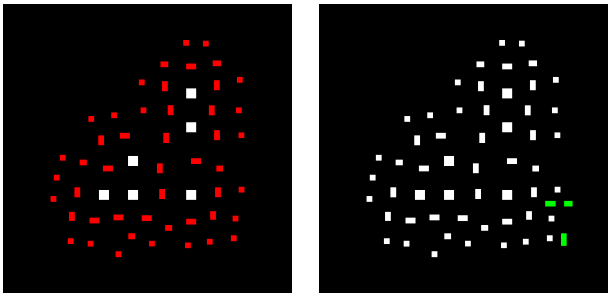
The work of [6] presents a realistic scenario of adversarial attacks on ML models targeting lithography hotspot detection. Currently, using a CNN-based hotspot detector, the designer can quickly ascertain if a layout with third-party macros is printable as-is. To pass off sub-par designs as high quality, 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, a stealthy poisoning attack is also threatening. It targets inserting backdoor in ML models during the training stage. Instead of requiring control over the model training process, this is achieved by poisoning the training data. A common poisoning mechanism is to insert a secret trigger to the features and coax ML models to unknowingly learn the secret trigger as a pattern of the attacker's target label. The work of [7] demonstrates poisoning attacks on lithography problems.

B. Attack Method on Model Performance

Adversarial attacks are based on the generation of adversarial samples. The most fundamental attack method is fast gradient sign attack (FGSM) [78]. For attacks without a specific target, it perturbs the input features X towards the direction that maximizes the error J between prediction $F(X|w)$ and the correct label y. This gradient ascent process is similar to the gradient descent operation on input in Equation 3, but optimizes input X towards the opposite direction. To avoid the attack being perceived by victims, the perturbation is often constrained with a maximum perturbation amount ϵ. For FGSM attack, the constraint ϵ is defined with an l_∞ norm. The generation process of perturbed input X_p is shown below.

$$X_p \leftarrow \text{clip}(X + \epsilon \ \text{sign}(\nabla_X J(F(X|w), y)))$$



(a) (b)

Fig. 4: Visualization of adversarial attacks targeting lithography problems. (a) The original input to ML hotspot detector. White shapes indicate vias and red shapes indicate SRAFs. (b) White shapes are original input (vias + SRAFs), green shapes are inserted artificial SRAF shapes (perturbations), which successfully fool the detector to switch from correct positive prediction (violation exists) to negative (no violation).

Besides the fundamental FGSM, there are other adversarial attack methods like projected gradient descent (PGD) [79], which is a more effective, multi-step variant of FGSM.

These fundamental attack methods with FGSM or PGD are based on constraints limiting pixel-wise perturbation amplitude, viewing input as ordinary images. In EDA applications, the scenario and attack algorithm can be quite different. For example, when targeting lithography hotspot detectors [6], instead of perturbing every pixel, the perturbation in this case is to insert or delete shapes of artificial sub-resolution assist features (SRAFs) on layouts. Also, the layout after perturbation has to remain legal and DRC violation clean [8], [6]. Compared with the well-studied traditional pixel-level attack, this largely different attack constraint leads to different attacks [8], [6] 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.

Figure 4 provides visualizations on attacks targeting lithography problems. Figure 4(a) shows the original model input with violations. The violation is correctly detected by the ML detector. Figure 4(b) shows the inserted perturbations that successfully fool the detector. Different from the traditional pixel-wise constraint, such perturbations have to be in the shape of SRAFs and the perturbed layout has to remain legal.

In ML for EDA, adversarial attacks are threatening to ML models targeting lithography problems, where design layouts as inputs can easily come from malicious third-party providers. In comparison, for models supposed to be deeply incorporated and coupled with existing design flows, like routability models, it will be more difficult for attackers to insert their perturbations to model inputs.

In addition, although we introduce adversarial attacks by assuming attackers have access to white-box model F with weights w, 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. This successful black-box attack attributes to the extraction of non-robust features by both surrogate model and target model F. Such non-robust features are features that are highly predictive, yet brittle and incomprehensible to humans [80]. Allowing black-box scenarios further lowers the bar for adversarial attacks on ML model performance.

Besides adversarial attacks, poisoning attacks target ML performance at 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 non-hotspot. 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 [7]. Recent studies [9] show that this poisoning attack on lithography can be defended by diluting the intentional bias from triggers with data augmentation strategies.

C. Defense Method on Model Performance

To cope with potential adversarial attacks in ML for EDA, we propose to build more robust models by adopting defense algorithms like curvature regularization (CURE) [81]. This work studies the relation between model *curvature* and robustness against adversarial attacks. It first calculates the Hessian matrix on loss function with respect to input features, then proves that ML models with a smaller curvature (i.e. smaller eigenvalues of the Hessian matrix) demonstrate higher robustness [81]. Intuitively, smaller eigenvalues of Hessian indicate a smaller curvature around input, implying a 'locally linear' behavior in the neighbor of input.

Therefore, to build more robust models, a solution is to penalize large eigenvalues of the aforementioned Hessian matrix with respect to the input. It is achieved by imposing gradient regularity (i.e., small curvature) along the direction of gradient descent. This new regularizer R with respect to original input X is shown below, and is added to the original loss function L with a controllable weight α.

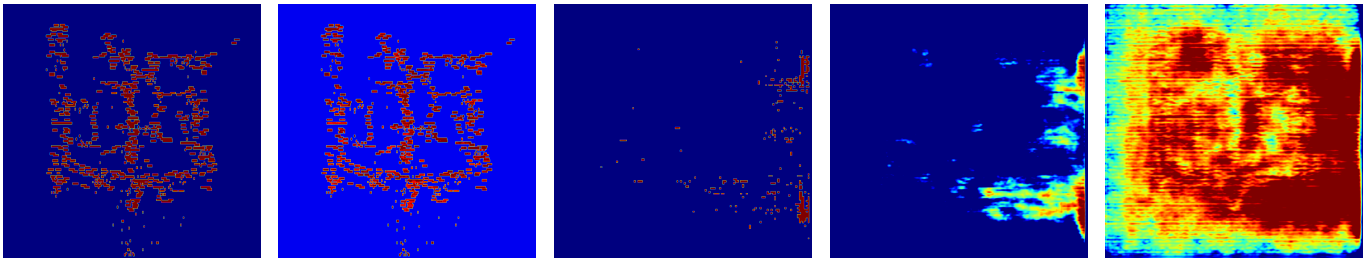
$$\begin{aligned} R &= ||\nabla_X L(X + hz) - \nabla_X L(X)||^2 \\ Loss &= L + \alpha R \end{aligned} \tag{5}$$

where the vector $z \propto \text{sign}(\nabla_X L(X))$, h is a sufficiently small value controlling the step size. This new regularizer R penalizes an approximation of the second-order derivative, which represents curvature with respect to input X.

In this paper, we studied adversarial attacks on different ML for EDA tasks. More importantly, we apply the CURE method to construct more robust models with very limited accuracy loss in tasks like lithography hotspot detection. It can better defend the adversarial attack [6] on hotspot detectors.

D. Experiment on Model Performance Attack

We first verify the effectiveness of the widely-adopted adversarial attack algorithm like PGD [79] with traditional



(a) (b) (c) (d) (e)

Fig. 5: Adversarial attack example on routability prediction. (a) Original feature on the distribution of clock tree. (b) The clock tree feature after adversarial attack. The perturbation is small. (c) The congestion label. (d) The prediction based on original features in a. (e) The prediction based on adversarial attacked features in b. The attack makes predictions meaningless.

Dataset Group	Model	Accuracy (AUC)	Attack Success Rate
1	Vanilla Model	0.829	0.338
	Robust Model	**0.842**	**0.200**
2	Vanilla Model	0.815	0.456
	Robust Model	**0.866**	**0.205**
3	Vanilla Model	0.741	0.338
	Robust Model	**0.800**	**0.235**
4	Vanilla Model	0.761	0.500
	Robust Model	**0.870**	**0.286**

TABLE IV: Adversarial attack and defense on lithography hotspot detectors. The attack inserts/deletes SRAFs on inputs. The robust model based on CURE regularizer reduces the attack success rate with limited accuracy loss.

pixel-wise constraint on routability models. These traditional adversarial attacks turn out to work well in ML for EDA tasks. Experimental results are shown in Figure 5. Figure 5(a) shows the original feature of the clock tree together with all flip-flops in the layout, and Figure 5(b) shows the corresponding feature with perturbations. Their difference in major patterns is not obvious. Figure 5(c) shows the congestion label. The normal prediction based on original features in Figure 5(d) is close to ground truth in Figure 5(c). However, the prediction based on features with perturbations in Figure 5(e) is almost meaningless. The distinction between prediction results in Figure 5(d) and Figure 5(e) clearly indicates the effectiveness of the traditional attacks like PGD in ML for EDA tasks, as demonstrated on this routability problem. Similar results are also observed for the FGSM attack in our experiment.

However, the difference between Figure 5(a) and 5(b) is still perceptual to humans, indicating inferior attack quality compared with attacks on general images. There are at least two reasons. First, the model performs binary classification on each grid instead of multi-class classification, leaving fewer inter-class decision boundaries. Second, the input feature is also close to binary, indicating the existence of the clock tree elements. The simple feature also makes perturbations more uniform and obvious.

As mentioned, adversarial attacks can be largely different in ML for EDA applications like lithography problems. As Figure 4 introduced, the corresponding adversarial attack constraint is different from the traditional pixel-wise attack. To study this type of attack, we first replicate the adversarial attack in the work of [6] with a similar experimental setup as mentioned in Subsection II-D. It attacks lithography hotspot detectors by inserting or deleting artificial SRAF shapes as perturbations. The accuracy of this model and attack success rate on it are shown in the 'vanilla model' of Table IV. In this experiment setup, there are four groups of data in the whole dataset. The attack success rate is measured by dividing the number of successful attacks by the total number of trials. After that, we apply the CURE regularizer in Equation 5 to construct a more robust model. As the comparison in Table IV shows, for group 1, the attack success rate drops from 0.338 in the vanilla model to 0.2 in our robust model, with the accuracy also slightly increasing from 0.829 to 0.842. This trend is the same in the other three dataset groups. It indicates the CURE-based robust model is less vulnerable to adversarial attacks in this specific task without any accuracy loss. This defense method can be directly applied during the model development process to generate a more robust model. In the future, we will further explore more robust models by customizing the CURE method to the constraint in SRAF shapes for this task.

VI. Unreliability in ML Performance

A. Model Unreliability Overview

We have discussed three major types of security concerns in ML for EDA, targeting data privacy, ML model competitive advantage, ML model performance, respectively. They are all malicious attacks. The last concern that is worth attention is model reliability. It is reflected by observations that model accuracy may seriously degrade on certain testing samples. It is not caused by any malicious attackers, but can be especially serious in ML for EDA compared with other ML applications, because of several special properties.

First, as mentioned in Subsection II-C, huge heterogeneity may exist between training and testing data samples, resulting from the large difference among circuits due to functionality, micro-architecture, and technology node. We further illustrate this concept in Figure 6, which shows a possible distribution of

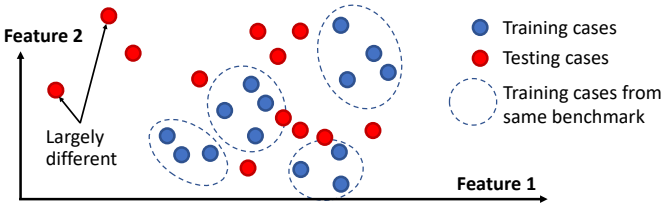


Fig. 6: Distribution of training and testing samples.

training and testing samples in a simplified input feature space. A few testing samples can be largely different from existing training samples, inevitably leading to accuracy degradation in these samples. In this example with only two features, the difference between samples seems obvious and easy to measure. But in practice, such 'difference' and the corresponding impact on testing accuracy are very difficult to know.

Second, it is very difficult for engineers to be aware of unexpected accuracy degradation on a few testing samples in practice. Accurate detection of accuracy degradation requires ground-truth labels to be collected, which is highly time-consuming and inherently against the purpose of adopting ML models. Undetected accuracy degradation can lead to much less optimized design solution or even chip failure, causing serious income loss for users from design companies.

Third, as mentioned in Subsection II-C, high complexity exists in both input data and the pattern for ML models to learn. These complexities exacerbate the difficulty in the study of input sample similarity or the detection of possible accuracy degradation.

In practice, the unreliability problem can be reflected by concerns like: 'Does the ML model work on 7nm technology or memory/GPU/certain IPs? To what extent may the accuracy degrade? Is transfer learning on new data necessary?' Currently this is mostly speculated based on model developers' confidence and intuition. 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 before they have a deep understanding of the potential unreliability in model performance. It affects all four scenarios we mentioned in Table I and may become a major obstacle that prevents a wide application of ML in EDA in the future.

B. Model Unreliability Analysis

For each ML model in EDA, understanding 'unreliability' requires detecting accuracy loss without knowing the label, or quantitatively determining the appropriate scope of testing samples. This solution is not straightforward. One direction we can think of is to define a new metric to measure the similarity between training data and each testing sample. As Figure 6 indicates, the performance unreliability (degradation) is mostly caused by the sparse distribution of data samples in the feature space. If the similarity between one testing sample and the model training data is lower than a certain threshold, the ML model should either reject inference on this testing sample or at least raise a warning. Another direction is to adopt ML models with prediction confidence incorporated in their prediction outputs. Low confidence generally indicates uncertainty and possible accuracy degradation on the testing sample. The confidence is available as probability values for many classifiers, especially multi-class classifiers, but less obvious in common regression tasks in ML for EDA.

Understanding 'unreliability' in ML models not only helps to avoid unexpected accuracy drop, but also provides guidance during model construction. If we can detect testing accuracy or define the appropriate scope of testing samples, then given a dataset, it is possible for developers to construct multiple ML models, each trained with part of training data and applied to a specific scope of testing samples. For example, in Figure 6, developers may train one ML model based on each benchmark, instead of training one general model with all samples in the training dataset. By applying different models to different testing samples, better overall results can be achieved.

C. Experiment on Unreliability and Data Similarity

We first demonstrate the accuracy degradation when applying models on largely different designs, and present our preliminary study in understanding the design similarity and model performance.

Accuracy degradation on specific testing samples is very common during the development of ML models. For example, while a carefully designed routability model on congestion prediction achieves an average performance of 0.83, its performance can be lower than 0.70 for a few testing designs.

To demonstrate the idea of measuring 'similarities' between design and samples, we try to visualize multiple layouts from various designs in different benchmarks in the routability experiment. We adopt simple principle component analysis (PCA) [82]-based dimension reduction techniques. The visualization is shown in Figure 7. Each point in this figure indicates one layout solution and same color indicate layouts from the same design. Their similarities in Euclidean distance can be directly visually observed in the figure. Different benchmark names are annotated on the figure. To provide more

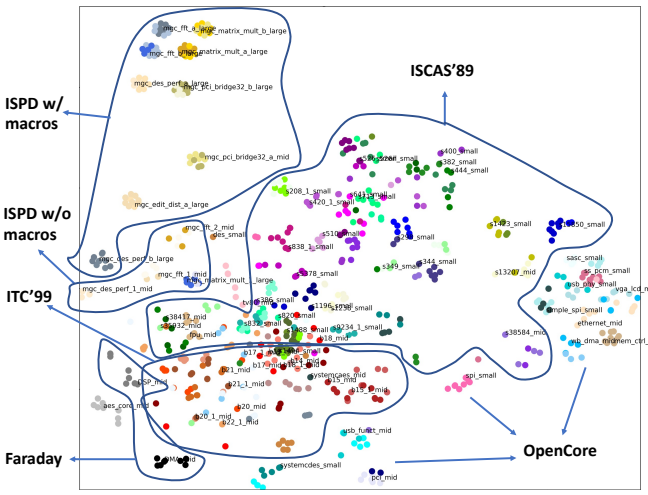


Fig. 7: Visualization of designs/layouts by dimension reduction. Each point represents one layout and same color indicates the same design. Similarities can be observed for designs/layouts from the same benchmark.

Test on	Training on		
	Middle	Small + Middle	Small + Middle + Large (All)
Small	70.6	**72.4**	71.5
Middle	**75.6**	75.4	71.3
Large	**71.3**	64.9	71.0

TABLE V: Testing accuracy on DRV detection. Three models are trained with different partitions of data. The model trained with all designs does not perform the best.

information, the tiny text annotated on some points is in the format of design name plus the size ('small', 'mid', 'large') of design. For example, the tiny text 'pci_mid' at bottom purple points of Figure 7 indicates design name 'pci' with middle-level layout size.

We can observe very interesting and reasonable clustering of layouts and designs in Figure 7. First, layouts from the same design with the same color are very closely clustered. Second, intuitively similar designs, like designs from the same benchmark, are obviously closer to each other. For example, all designs from the ISPD benchmark distribute on the upper left corner of Figure 7. More importantly, the designs with macros are clearly closer to the corner, showing a larger difference with most designs without macros. Similarly, designs from ITC'99 and small designs from ISCAS'89 reflect clear intra-benchmark similarities. It is highly likely such straightforward similarity measurement is not the best solution, but it already demonstrates reasonable patterns. Perhaps a better solution should capture more similarities in local patterns and gate connection topologies. Studies like this can provide guidance in quantitative measurements of design similarity and understanding of model unreliability. A straightforward example is, models trained with small design layouts (in the center of Figure 7) may not perform well on large designs with macros (in the upper left corner).

As mentioned, such design similarity also provides guidance in model construction. We provide an experiment on developing DRV detection models with different training data. Notice that this preliminary experiment targets DRV, thus overall accuracy tends to be lower than congestion detection in previous experiments. In this experiment, all training and testing designs are classified into three types: small, middle, large, according to their layout size. This layout size is another highly straightforward indication of design similarity and we believe a huge room for improvement exists. Based on this partition of data, we try to train models either on all training data or on part of training data. We report the accuracies of different models on different testing data in Table V.

As Table V shows, the model trained on all designs does not perform the best. Instead, the model trained only on middle designs performs better on middle and large testing designs, as shown in the last two rows in Table V. In other words, including small designs in training data generally degrades the model's performance on middle and large designs. Intuitively it can be explained by the 'difference' between small designs and middle or large designs.

Similarly, when testing on small designs, as shown in the first row of Table V, including small designs in training improves accuracy while including large designs degrades accuracy. Generally, training with designs in similar sizes tends to yield a better model, which is intuitively reasonable. This preliminary result supports our speculation that based on design similarity, constructing multiple ML models for different testing scopes can achieve better accuracy.

In summary, preliminary studies in Figure 7 and Table V indicate that a good utilization of data similarity may help mitigate model unreliability problems caused by the data heterogeneity. During model development, multiple sub-models can be generated with different partitions of training data. When applying ML models, each testing sample may be evaluated to determine whether inference should be rejected due to uncertainty or which sub-model to apply, before the prediction result is generated.

VII. Potential Concerns in the Future

We have presented four major types of concerns in ML for EDA. At the end of this study, we try to further anticipate a few other potential concerns or impacts that may arise in the future, when ML for EDA becomes more ubiquitous.

A. Security in Decentralized Setting

The effectiveness of ML for EDA largely hinges on the availability of a large amount of high-quality training data. In reality, developers have very limited access to the latest design data, which is owned by design companies and mostly confidential. Such data availability problem is becoming the limiting constraint on the future growth of ML for EDA and chip design. Considering the decentralized distribution of high-quality circuit data, we have witnessed explorations [59] based on federated learning (FL), as Figure 8 illustrates. Developers collaboratively train one ML model based on the private local data from K data providers. In each round, data providers send locally trained models to the central server, then the server aggregates and distributes the updated model back to all providers. This may become a major trend in constructing and deploying ML models in the future.

However, collaboratively constructing ML models in a decentralized setting incurs many new security concerns. For example, if one of the data providers is an attacker, it leads to serious security threats. First, the attacker can directly get full access to the trained ML model during the collaborative training process. Second, the attacker can easily insert malicious backdoor attacks into the ML model, by including poisoning samples in its local training dataset. Third, it is possible for the attacker to recover the private data of other data providers. This can be achieved based on the idea of generative adversarial networks (GAN). The attacker can use the trained model as a discriminator, and train an additional generator to recover input samples of a specific class [83]. But this requires the model to be a multi-class classifier, which is not common in ML for EDA tasks.

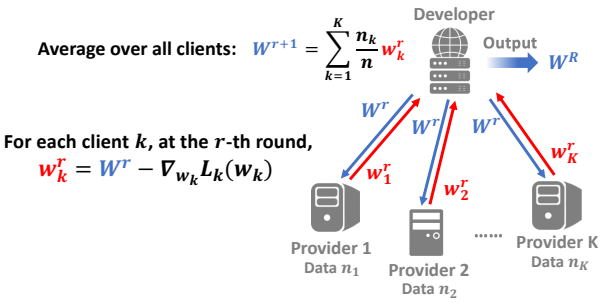


Fig. 8: The visualization of the decentralized training.

B. Label Generation with ML Models

As mentioned in Subsection IV-B, label generation is one of the most costly steps in model development. In the future when ML models for EDA achieve higher accuracy and become ubiquitous, it is possible to directly apply existing models to generate training labels for the development of new ML models. While this greatly reduces label generation cost, accuracy degradation is unavoidable.

The accuracy degradation can already be observed in the model extraction experiment in Subsection IV-C. As Table III shows, the attack model trained on 40% unlabeled data is less accurate than the original victim model trained on 40% labeled data. Model developers should be aware of such accuracy loss and avoid overuse of pseudo-labels.

C. Impact on EDA Tools

In the future, models learning the functionality of EDA tools may be applied to partially or even entirely replace some functions in these EDA tools in circuit design flow. This may seem quite impractical to many practitioners. However, there exist works [25], especially reinforcement learning-based methods, have already claimed superior performance over the same function in existing tools without reliance on them in applications. Although debates still exist on the actual performance of these current solutions, such possibilities may not be ruled out in our anticipation. Different from most ML applications where models replace human efforts, ML for EDA methods have been applied to accomplish the tasks of both human designers and EDA tools.

To avoid emerging competition with their own tools, in the future EDA vendors may hope to revise existing user license agreements and prevent unauthorized use of their tools to develop ML models with similar functionalities. However, disabling model training based on a specific software is technically very difficult, and violations of this rule cannot be easily detected by the vendor.

VIII. Conclusion

In this paper, we provide a comprehensive and impartial summary of all security and reliability concerns we observe in ML for EDA tasks. According to our study, some concerns like model extraction, attacks on model performance, and inherent model unreliability are highly threatening, while potential design privacy attack turns out to be less practical. In the future, we will explore more customized attack and defense methods with more in-depth experiments in ML for EDA.

References

[1] IBS, "As chip design costs skyrocket, 3nm process node is in jeopardy," https://www.extremetech.com/computing/272096-3nm-process-node, 2020.

[2] G. Huang, J. Hu, Y. He, J. Liu, M. Ma, Z. Shen, J. Wu, Y. Xu, H. Zhang, K. Zhong *et al.*, "Machine learning for electronic design automation: A survey," *arXiv preprint arXiv:2102.03357*, 2021.

[3] M. Rapp, H. Amrouch, Y. Lin, B. Yu, D. Z. Pan, M. Wolf, and J. Henkel, "MLCAD: A survey of research in machine learning for CAD keynote paper," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2021.

[4] Synopsys, "DSO.ai: Ai-driven design applications," 2021. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/ml-ai-design/dso-ai.html

[5] Cadence, "Cadence Cerebrus intelligent chip explorer," 2021. [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/cerebrus-intelligent-chip-explorer.html

[6] K. Liu, H. Yang, Y. Ma, B. Tan, B. Yu, E. F. Young, R. Karri, and S. Garg, "Adversarial perturbation attacks on ML-based cad: A case study on CNN-based lithographic hotspot detection," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 2020.

[7] K. Liu, B. Tan, R. Karri, and S. Garg, "Poisoning the (data) well in ML-based CAD: A case study of hiding lithographic hotspots," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[8] H. Yang, S. Zhang, K. Liu, S. Liu, B. Tan, R. Karri, S. Garg, B. Yu, and E. F. Young, "Attacking a CNN-based layout hotspot detector using group gradient method," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021.

[9] K. Liu, B. Tan, G. R. Reddy, S. Garg, Y. Makris, and R. Karri, "Bias busters: Robustifying DL-based lithographic hotspot detectors against backdooring attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2020.

[10] S. Dai, Y. Zhou, H. Zhang, E. Ustun, E. F. Young, and Z. Zhang, "Fast and accurate estimation of quality of results in high-level synthesis with machine learning," in *Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2018.

[11] H. M. Makrani, F. Farahmand, H. Sayadi, S. Bondi, S. M. P. Dinakarrao, H. Homayoun, and S. Rafatirad, "Pyramid: Machine learning framework to estimate the optimal timing and resource usage of a high-level synthesis design," in *International Conference on Field-Programmable Logic and Applications (FPL)*, 2019.

[12] H.-Y. Liu and L. P. Carloni, "On learning-based methods for design-space exploration with high-level synthesis," in *Design automation conference*, 2013.

[13] D. Liu and B. C. Schafer, "Efficient and reliable high-level synthesis design space explorer for FPGAs," in *International Conference on Field Programmable Logic and Applications (FPL)*, 2016.

[14] Y. Zhou, H. Ren, Y. Zhang, B. Keller, B. Khailany, and Z. Zhang, "PRIMAL: Power inference using machine learning," in *Design Automation Conference (DAC)*, 2019.

[15] Y. Zhang, H. Ren, and B. Khailany, "GRANNITE: Graph neural network inference for transferable power estimation," in *Design Automation Conference (DAC)*, 2020.

[16] Z. Xie, X. Xu, M. Walker, J. Knebel, K. Palaniswamy, N. Hebert, J. Hu, H. Yang, Y. Chen, and S. Das, "Apollo: An automated power modeling framework for runtime power introspection in high-volume commercial microprocessors," in *International Symposium on Microarchitecture (MICRO)*, 2021.

[17] D. Kim, J. Zhao, J. Bachrach, and K. Asanović, "Simmani: Runtime power modeling for arbitrary rtl with automatic signal selection," in *International Symposium on Microarchitecture (MICRO)*, 2019.

[18] C. Yu, H. Xiao, and G. De Micheli, "Developing synthesis flows without human knowledge," in *Design Automation Conference (DAC)*, 2018.

[19] W. L. Neto, M. Austin, S. Temple, L. Amaru, X. Tang, and P.-E. Gaillardon, "LSOracle: A logic synthesis framework driven by artificial intelligence," in *International Conference On Computer Aided Design (ICCAD)*, 2019.

[20] A. Hosny, S. Hashemi, M. Shalan, and S. Reda, "Drills: Deep reinforcement learning for logic synthesis," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[21] W. L. Neto, M. T. Moreira, Y. Li, L. Amarù, C. Yu, and P.-E. Gaillardon, "Slap: A supervised learning approach for priority cuts technology mapping," in *Design Automation Conference (DAC)*, 2021.

[22] E. C. Barboza, N. Shukla, Y. Chen, and J. Hu, "Machine learning-based pre-routing timing prediction with reduced pessimism," in *Design Automation Conference (DAC)*, 2019.

[23] A. B. Kahng, M. Luo, and S. Nath, "Si for free: machine learning of interconnect coupling delay and transition effects," in *International Workshop on System Level Interconnect Prediction (SLIP)*, 2015.

[24] Z. Xie, R. Liang, X. Xu, J. Hu, C.-C. Chang, J. Pan, and Y. Chen, "Pre-placement net length and timing estimation by customized graph neural network," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2022.

[25] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi *et al.*, "A graph placement methodology for fast chip design," *Nature*, 2021.

[26] Y.-H. Huang, Z. Xie, G.-Q. Fang, T.-C. Yu, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "Routability-driven macro placement with embedded CNN-based prediction model," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019.

[27] Z. Xie, Y.-H. Huang, G.-Q. Fang, H. Ren, S.-Y. Fang, Y. Chen, and J. Hu, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *International Conference On Computer Aided Design (ICCAD)*, 2018.

[28] C. Yu and Z. Zhang, "Painting on placement: Forecasting routing congestion using conditional generative adversarial nets," in *Design Automation Conference (DAC)*, 2019.

[29] J. Chen, J. Kuang, G. Zhao, D. J.-H. Huang, and E. F. Young, "PROS: A plug-in for routability optimization applied in the state-of-the-art commercial eda tool using deep learning," in *International Conference On Computer Aided Design (ICCAD)*, 2020.

[30] Z. Xie, H. Ren, B. Khailany, Y. Sheng, S. Santosh, J. Hu, and Y. Chen, "PowerNet: Transferable dynamic IR drop estimation via maximum convolutional neural network," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[31] C.-T. Ho and A. B. Kahng, "Incpird: Fast learning-based prediction of incremental ir drop," in *International Conference on Computer-Aided Design (ICCAD)*, 2019.

[32] H. Zhou, W. Jin, and S. X.-D. Tan, "Gridnet: Fast data-driven em-induced ir drop prediction and localized fixing for on-chip power grid networks," in *International Conference On Computer Aided Design (ICCAD)*, 2020.

[33] Y.-C. Lu, J. Lee, A. Agnesina, K. Samadi, and S. K. Lim, "GAN-CTS: A generative adversarial framework for clock tree prediction and optimization," in *International Conference On Computer Aided Design (ICCAD)*, 2019.

[34] Z. Xie, R. Liang, X. Xu, J. Hu, Y. Duan, and Y. Chen, "Net^2: A graph attention network method customized for pre-placement net length estimation," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2021.

[35] R. Liang, Z. Xie, J. Jung, V. Chauha, Y. Chen, J. Hu, H. Xiang, and G.-J. Nam, "Routing-free crosstalk prediction," in *International Conference on Computer Aided Design (ICCAD)*, 2020.

[36] Y.-C. Lu, S. S. K. Pentapati, L. Zhu, K. Samadi, and S. K. Lim, "TP-GNN: A graph neural network framework for tier partitioning in monolithic 3d ICs," in *Design Automation Conference (DAC)*, 2020.

[37] Y. Katz, M. Rimon, A. Ziv, and G. Shaked, "Learning microarchitectural behaviors to improve stimuli generation quality," in *Design Automation Conference (DAC)*, 2011.

[38] S. Fine and A. Ziv, "Coverage directed test generation for functional verification using bayesian networks," in *Design Automation Conference (DAC)*, 2003.

[39] Y. Ma, H. Ren, B. Khailany, H. Sikka, L. Luo, K. Natarajan, and B. Yu, "High performance graph convolutional networks with applications in testability analysis," in *Design Automation Conference (DAC)*, 2019.

[40] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2018.

[41] H. Yang, S. Li, Z. Deng, Y. Ma, B. Yu, and E. F. Young, "GAN-OPC: Mask optimization with lithography-guided generative adversarial nets," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2019.

[42] J. Kwon, M. M. Ziegler, and L. P. Carloni, "A learning-based recommender system for autotuning design flows of industrial high-performance processors," in *Design Automation Conference (DAC)*, 2019.

[43] Z. Xie, G.-Q. Fang, Y.-H. Huang, H. Ren, Y. Zhang, B. Khailany, S.-Y. Fang, J. Hu, Y. Chen, and E. C. Barboza, "Fist: A feature-importance sampling and tree-based method for automatic design flow parameter tuning," in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2020.

[44] K. Kunal, J. Poojary, T. Dhar, M. Madhusudan, R. Harjani, and S. S. Sapatnekar, "A general approach for identifying hierarchical symmetry constraints for analog circuit layout," in *International Conference On Computer Aided Design (ICCAD)*, 2020.

[45] H. Li, F. Jiao, and A. Doboli, "Analog circuit topological feature extraction with unsupervised learning of new sub-structures," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016.

[46] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han, "GCN-RL circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning," in *Design Automation Conference (DAC)*, 2020.

[47] K. Hakhamaneshi, N. Werblun, P. Abbeel, and V. Stojanović, "Bagnet: Berkeley analog generator with layout optimizer boosted with deep neural networks," in *International Conference on Computer-Aided Design (ICCAD)*, 2019.

[48] H. Ren, G. F. Kokai, W. J. Turner, and T.-S. Ku, "ParaGraph: Layout parasitics and device parameter prediction using graph neural networks," in *Design Automation Conference (DAC)*, 2020.

[49] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar, "GANA: Graph convolutional network based automated netlist annotation for analog circuits," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[50] M. Liu, K. Zhu, J. Gu, L. Shen, X. Tang, N. Sun, and D. Z. Pan, "Towards decrypting the art of analog layout: Placement quality prediction via transfer learning," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2020.

[51] Y. Li, Y. Lin, M. Madhusudan, A. Sharma, W. Xu, S. S. Sapatnekar, R. Harjani, and J. Hu, "A customized graph neural network model for guiding analog ic placement," in *International Conference On Computer Aided Design (ICCAD)*, 2020.

[52] B. Xu, Y. Lin, X. Tang, S. Li, L. Shen, N. Sun, and D. Z. Pan, "Wellgan: Generative-adversarial-network-guided well generation for analog/mixed-signal circuit layout," in *Design Automation Conference (DAC)*, 2019.

[53] K. Zhu, M. Liu, Y. Lin, B. Xu, S. Li, X. Tang, N. Sun, and D. Z. Pan, "GeniusRoute: A new analog routing paradigm using generative neural network guidance," in *International Conference on Computer-Aided Design (ICCAD)*, 2019.

[54] H.-G. Stratigopoulos and Y. Makris, "Error moderation in low-cost machine-learning-based analog/rf testing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, 2008.

[55] Cadence, "Innovus implementation system," 2021. [Online]. Available: https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html

[56] Synopsys, "IC Compiler II for physical implementation," 2021. [Online]. Available: https://www.synopsys.com/implementation-and-signoff/physical-implementation/ic-compiler.html

[57] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *USENIX Security Symposium*, 2016.

[58] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, 2016.

[59] J. Pan, C.-C. Chang, Z. Xie, A. Li, M. Tang, T. Zhang, J. Hu, and Y. Chen, "Towards collaborative intelligence: Routability estimation based on decentralized private data," in *Design Automation Conference (DAC)*, 2022.

[60] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017.

[61] P. Yanardag and S. Vishwanathan, "Deep graph kernels," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.

[62] A. Salem, Y. Zhang, M. Humbert, P. Berrang, M. Fritz, and M. Backes, "Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models," *arXiv preprint arXiv:1806.01246*, 2018.

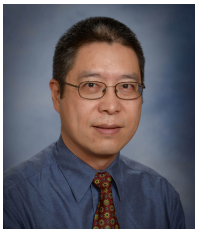
[63] C.-C. Chang, J. Pan, T. Zhang, Z. Xie, J. Hu, W. Qi, C. Lin, R. Liang, J. Mitra, E. Fallon, and Y. Chen, "Automatic routability predictor development using neural architecture search," in *International Conference On Computer Aided Design (ICCAD)*, 2021.

[64] R. Liang, H. Xiang, D. Pandey, L. Reddy, S. Ramji, G.-J. Nam, and J. Hu, "DRC hotspot prediction at sub-10nm process nodes using customized convolutional network," in *International Symposium on Physical Design (ISPD)*, 2020.
[65] F. Brglez, D. Bryan, and K. Kozminski, "Combinational profiles of sequential benchmark circuits," in *International Symposium on Circuits and Systems (ISCAS)*, 1989, pp. 1929–1934.
[66] F. Corno, M. S. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *Design & Test of computers*, 2000.
[67] C. Albrecht, "IWLS 2005 benchmarks," in *IWLS: http://www.iwls.org*, 2005.
[68] I. S. Bustany, D. Chinnery, J. R. Shinnerl, and V. Yutsis, "ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement," in *International Symposium on Physical Design (ISPD)*, 2015.
[69] Si2, "NanGate 45nm Open Cell Library," 2018. [Online]. Available: https://si2.org/open-cell-library/
[70] N. Carlini, C. Liu, Ú. Erlingsson, J. Kos, and D. Song, "The secret sharer: Evaluating and testing unintended memorization in neural networks," in *USENIX Security Symposium*, 2019.
[71] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Conference on Computer and Communications Security (CCS)*, 2015.
[72] H. Yin, P. Molchanov, J. M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N. K. Jha, and J. Kautz, "Dreaming to distill: Data-free knowledge transfer via DeepInversion," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.
[73] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning (ICML)*, 2015.
[74] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *Advances in neural information processing systems*, vol. 32, 2019.
[75] S. J. Oh, M. Augustin, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *International Conference on Learning Representations (ICLR)*, 2018.
[76] M. Rigaki and S. Garcia, "A survey of privacy attacks in machine learning," *arXiv preprint arXiv:2007.07646*, 2020.
[77] M. Juuti, S. Szyller, S. Marchal, and N. Asokan, "Prada: protecting against dnn model stealing attacks," in *European Symposium on Security and Privacy (EuroS&P)*, 2019.
[78] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.
[79] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.
[80] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," *Advances in Neural Information Processing Systems (NeurIPs)*, 2019.
[81] S.-M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard, "Robustness via curvature regularization, and vice versa," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
[82] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and Intelligent Laboratory Systems*, 1987.
[83] B. Hitaj, G. Ateniese, and F. Perez-Cruz, "Deep models under the GAN: information leakage from collaborative deep learning," in *Conference on Computer and Communications Security (CCS)*, 2017.

Zhiyao Xie received the Ph.D. degree in computer engineering from Duke University in 2022, and B.E. degree in electronic engineering from the City University of Hong Kong in 2013. He is now an Assistant Professor of the Department of Electronic and Computer Engineering at the Hong Kong University of Science and Technology (HKUST). His research interests include machine learning and its applications in EDA, VLSI design, and computer architecture. He received the Best Paper Award in MICRO 2021.

Jingyu Pan received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 2020. He is currently a Ph.D. student in the Electrical and Computer Engineering department at Duke University. His research interests include machine learning applications in Electronics Design Automation and VLSI circuits and systems.

Chen-Chia Chang is a Ph.D. student in Electrical and Computer Engineering at Duke University. He is advised by Prof. Yiran Chen and Prof. Helen Li in the Computational Evolutionary Intelligence Lab. His research interests are focused on Electronic Design Automation (EDA) and machine learning algorithms. He received a B.S. (2020) in Electrical Engineering at National Taiwan University.



Jiang Hu (Fellow, IEEE) received the Ph.D. degree from the University of Minnesota, Minneapolis, MN, USA, in 2001. He has worked with IBM Microelectronics, Armonk, NY, USA, from 2001 to 2002, and has been a Faculty Member with Texas A&M University, College Station, TX, USA. Dr. Hu received Best Paper Awards at ACM/IEEE Design Automation Conference 2001, IEEE/ACM International Conference on Computer-Aided Design 2011, IEEE International Conference on Vehicular Electronics and Safety 2018, and IEEE/ACM International Symposium on Microarchitecture 2021. He also received the IBM Invention Achievement Award and the Humboldt Research Fellowship. He has served on the editorial boards of the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and the ACM Transactions on Design Automation of Electronic Systems. He was the Technical Program Chair of the ACM International Symposium on Physical Design 2011.

Yiran Chen received B.S (1998) and M.S. (2001) from Tsinghua University and Ph.D. (2005) from Purdue University. After five years in industry, he joined University of Pittsburgh in 2010 as Assistant Professor and then was promoted to Associate Professor with tenure in 2014, holding Bicentennial Alumni Faculty Fellow. He is now the Professor of the Department of Electrical and Computer Engineering at Duke University and serving as the director of the NSF AI Institute for Edge Computing Leveraging the Next-generation Networks (Athena), the NSF Industry–University Cooperative Research Center (IUCRC) for Alternative Sustainable and Intelligent Computing (ASIC), and the co-director of Duke Center for Computational Evolutionary Intelligence (DCEI). His group focuses on the research of new memory and storage systems, machine learning and neuromorphic computing, and mobile computing systems. Dr. Chen has published 1 book and about 500 technical publications and has been granted 96 US patents. He has served as the associate editor of more than a dozen international academic periodicals and served on the technical and organization committees of more than 60 international conferences. He is now serving as the Editor-in-Chief of the IEEE Circuits and Systems Magazine. He received eight best paper awards, one best poster award, and fourteen best paper nominations from reputable international conferences and workshops such as MICRO, KDD, DATE, SEC, etc. He received numerous awards for his technical contributions and professional services such as IEEE Computer Society Edward J. McCluskey Technical Achievement Award, ACM SIGDA Service Award, etc. He is a Fellow of the ACM and IEEE and now serves as the chair of ACM SIGDA.