Rethink before Releasing your Model: ML Model Extraction Attack in EDA

Chen-Chia Chang[1], Jingyu Pan[1], Zhiyao Xie[2], Jiang Hu[3], and Yiran Chen[1]
Duke University[1] Hong Kong University of Science and Technology[2] Texas A&M University[3]
{chenchia.chang, jingyu.pan, yiran.chen}@duke.edu, eezhiyao@ust.hk, jianghu@tamu.edu

ABSTRACT

Machine learning (ML)-based techniques for electronic design automation (EDA) have boosted the performance of modern integrated circuits (ICs). Such achievement makes ML model to be of importance for the EDA industry. In addition, ML models for EDA are widely considered having high development cost because of the time-consuming and complicated training data generation process. Thus, confidentiality protection for EDA models is a critical issue. However, an adversary could apply *model extraction attacks* to *steal* the model in the sense of achieving the comparable performance to the victim's model. As model extraction attacks have posed great threats to other application domains, e.g., computer vision and natural language process, in this paper, we study model extraction attacks for EDA models under two real-world scenarios. It is the first work that (1) introduces model extraction attacks on EDA models and (2) proposes two attack methods against the unlimited and limited query budget scenarios. Our results show that our approach can achieve competitive performance with the well-trained victim model without any performance degradation. Based on the results, we demonstrate that model extraction attacks truly threaten the EDA model privacy and hope to raise concerns about ML security issues in EDA.

1 INTRODUCTION

Modern IC design development involves a multi-stage design flow. However, with limited interplay across different stages, tools in the early stage cannot guarantee high-quality solutions for subsequent stages, causing the need of many optimization iterations. As technology node shrinks, this drawback leads to an even longer turnaround time in IC design development. Thus, new EDA methodologies for design efficiency improvement is in a high demand.

ML techniques, which have shown great ability in prediction and optimization, play important roles in the advanced EDA tool development. A wide range of ML-based approaches [11] is proposed to foresee circuit quality across different stages, such as routability prediction [7, 8, 17, 28, 31] and lithographic hotspot detection [18, 30]. In addition, many industrial tools [5, 25] have demonstrated great potential in integrating ML models into EDA design flows to achieve better IC power, performance, and area (PPA). The achievement exhibits the power of ML techniques, leading to the high demand of ML for EDA.

With the fast development of ML for EDA techniques, the next important phase for their wide adoption is about the commercialization. An essential question to consider is, what will be the possible business scenarios of ML for EDA? We envision two possibilities based on our understanding. First, some giant semiconductor companies may be able to develop and use ML models all by themselves. However, this scenario is hard to be supported by small or even middle-class companies because its development requires expertise with a diverse background, including ML, EDA, and IC design. Thus, a more 'democratic' scenario that may benefit more IC designers could be separate ML model vendors and users. This specialization would be more convenient for those vendors/users to focus on developing/applying ML models for EDA. We believe this scenario is practicable since it is also observed in other domain applications. For example, Microsoft Azure [19] provides natural language processing models through web Application Programming Interfaces (APIs) in a secure cloud platform to support queries from users. In this way, they can choose not to disclose its model to users and thus protect their model, which has great business value. Similarly, we expect ML models for EDA can also be provided as a service through APIs, and we focus on such separate-vendor-user scenario in this paper.

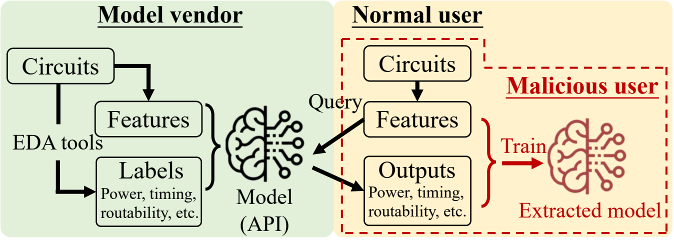


Figure 1: An anticipated application scenario of ML for EDA model and the corresponding model extraction attack. The vendor trains a model with circuit features and labels generated by EDA tools. Users are expected to take their own circuit features as inputs to query their circuits' quality. However,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

The details of this envisioned scenario is depicted in Figure 1. The vendor first trains its ML model using circuit features and labels produced by EDA tools. Such ML model could be mostly used to perform quality prediction (e.g., power, timing, and routability). This well-trained model, which is also referred to as an *oracle*, is then accessed by users as a black box. Therefore, by feeding the circuit features into the oracle, normal users can obtain the quality prediction results and then use results to facilitate circuit development in the subsequent stages.

However, in this scenario, the business value of the vendor's model is under risk because the model could be replicated by malicious users with *model extraction attacks* [13, 15, 20]. In such attacks, malicious users aim to *steal* or *replicate* the remotely deployed model with the goal of achieving the competitive performance. In the scenario sketched in Figure 1, malicious users can easily build their own substitute models by exploiting the vendor-provided oracle. They can obtain abundant *pseudo labels* in a short period and train models without the label generation process with EDA tools. In ML for EDA, label generation is a burdensome task. As the example in Table 1 shows, obtaining only one post-routing label for a moderate-size design (with 100k nets) requires approximately 2 hours by running the commercial tool. Therefore, constructing a dataset with thousands of data points would take up to several months for the vendors. In comparison, obtaining a label through an existing ML model only takes 0.03 seconds. Thus, malicious users can save lots of model development time by utilizing the oracle. As a result, such model extraction attacks will greatly hurt vendor's

ASPDAC '23, January 16–19, 2023, Tokyo, Japan
© 2023 Association for Computing Machinery.
ACM ISBN 978-1-4503-9783-4/23/01…$15.00
https://doi.org/10.1145/3566097.3567896

Table 1: Post-routing label generation time. To generate labels for the ITC'99 [9] benchmark circuit b_19 with 100k nets, commercial EDA tool requires approximately 2.22 hours while the well-known ML model [28] takes 0.03 seconds.

Label generation method	Runtime
Routing with EDA tool (Innovus® [6])	2.22 hr
ML model [28]	0.03 sec

business advantage. This undesirable consequence may hamper the development and commercialization of ML techniques for EDA.

To validate the attack concern, we investigate model extraction techniques in ML for EDA. In other domain applications, there has been some previous explorations [13, 15, 20]. However, they cannot be directly applied in EDA tasks. For example, a well-known work [20] tries to extract a class-specific classifier from a multi-class model. However, its goal is different to ours because our attacks aim to extract the exact same model as the oracle. Second, the work [13] utilizes the rotation loss to robustify pseudo-label training, while the rotation loss is not applicable for most tasks on circuit layouts because features and labels could be mismatched after rotation. Finally, the other work [15] attacks models that input word sequence, while such word sequence is largely different from the data format of circuits. Given the uniqueness of EDA problems, we need to study new methods for model extraction attacks.

To the best of our knowledge, we are the first to study the potential threat of model extraction attack in EDA. To study its threat, we construct real-world scenarios targeting routability prediction [7, 8, 17, 28, 31], which is a well-studied ML application for EDA. In this case study, the attacker's goal is to construct a new routability model that can achieve competitive performance with the vendor's one. Given no ground-truth labels to the attacker, a naive model extraction attack is to use pseudo labels from the oracle to train a model. However, the attack model's performance cannot be guaranteed because the oracle may produce false labels due to the model imperfection with unseen data, which could actually mislead the attack model. Also, the attacker could have limited query budgets since model vendors may charge users per query or set a hard limit to prevent potential extraction attacks. In this work, we propose two more effective model extraction methods by selecting reliable and high-informative pseudo labels to validate the threat of model extraction attacks.

Our contributions are summarized as follows:

- We raise the concern of model extraction attacks in ML for EDA and validate that such attacks truly jeopardize the development and the commercialization of ML models for EDA through our results. In this study, we employ a realistic attack setup that the attacker has no prior knowledge about the vendor's model structure and ground-truth labels.
- To examine the threat of model extraction attacks in EDA, we propose two effective attack methods. Without query budgets, we propose a confidence-based data selection method to avoid attacker model being misguided by the unreliability pseudo labels. With limited query budgets, we propose an information-based iterative data selection method to progressively choose the most informative pseudo labels to the attacker model and apply self-training to improve the model generalization ability.
- Without query budgets, our method can reach competitive performance to the oracle without any performance degradation and outperform a naive attack method with 5.0% ROC-AUC improvement. With limited query budgets, our method can outperform simple random selection with average 3.4% improvement under varied query budget.

Through this article, we explore the risks of model commercialization imposed by model extraction attacks and call attentions to more ML security issues [29] for EDA in future studies.

The reminder of the paper is organized as follows. Section 2 details the preliminaries, including our target application and two EDA model extraction scenarios. For each scenario, we propose a data selection method to optimize the attack performance. For the first scenario, Section 3 introduces the confidence-based data selection method. For the second scenario, Section 4 gives the information-based iterative data selection method. Then, Section 5 demonstrates the experimental results, and Section 6 concludes our paper.

2 PRELIMINARIES

In this section, we first describe our target EDA application, the routability prediction. Then, we define real-world model extraction scenarios for EDA.

2.1 Target application: Routability prediction

As a proof of concept demonstration, we apply model extraction attacks on the routability prediction application [7, 8, 17, 28, 31, 32]. This ML-based estimation is widely applied to improve placers [12] and routers [8], making great contribution to the physical design flow. Therefore, we select it as our target application. Routability prediction estimates the routability of a placement result based on the routing congestion occurred in global routing. Possessing the routability prediction model, we can examine whether the current placement result can lead to the congestion-free routing solution in the later routing stage. Routability prediction can be formally defined as follows: Given a set of placement solutions with the features X, a routability prediction model f takes X as inputs to predict routing congestion Y, where

$$f : X_i \in \mathbb{R}^{w\times h\times c} \rightarrow Y_i \in \mathbb{B}^{w\times h}.$$

In this equation, w and h are the width and height of the placement result, c is the number of features, and $\mathbb{B}$ indicates whether this location has congestion.

2.2 Model Extraction Scenarios

In this section, we detail two representative scenarios for EDA model extraction attacks and define the corresponding problem formulations.

In the scenario, there are an ML model vendor and a malicious user (attacker). The well-trained model f_v provided by the vendor is referred to as the *victim model* and can be queried by users as an oracle to provide prediction results. As a model buyer, the attacker has a set of circuits C, which generates multiple unlabeled layouts and transforms into features X_i to be predicted, and we refer this set of features for the circuit c as $\mathbb{U}_c = \{X_i | X_i \in c\}$. The whole unlabeled dataset is referred to as $\mathbb{U} = \{X_i\}_{i=1}^{u} = \{\bigcup_{c=1}^{|C|} U_c\}$. The attacker's objective is to construct a model f_a, which is referred to as the *attacker model*, to achieve the competitive or even better performance to f_v. Based on this scenario, the problem formulation for the model extraction attack can be defined as follows:

Problem 1 (**Application scenario 1**). *Given a well-trained black-box victim model f_v and an attacker's unlabeled dataset $\mathbb{U}$, the objective is to build an attacker model f_a such that the performance of f_a evaluated on $\mathbb{U}$ is maximized.*

In this paper, we further consider the scenario that the attacker has a query budget. For example, model vendors may deploy ML models as web APIs to charge users per query. Thus, users could only have limited queries due to finite budgets. This scenario can be formulated as follows:

Problem 2 (**Application scenario 2**). *Given a well-trained black-box victim model f_v, an attacker's unlabeled dataset $\mathbb{U}$, and a query budget β, the objective is to build an attacker model f_a such that the performance of f_a evaluated on $\mathbb{U}$ is maximized.*

Algorithm 1 *Confidence-based data selection*

```
Input: Victim model $f_v$, unlabeled dataset $\mathbb{U}$
Output: Attacker model $f_a$ with weights $w_a$
 1: $\mathbb{P} = \{(X, f_v(X)) | X \in \mathbb{U}\}$
 2: $w_a = \text{train}(f_a, \mathbb{P})$
 3: mc_array = $\emptyset$
 4: for $X_i \in \mathbb{U}$ do
 5:     $MC_i = \text{MC}(X_i, w_a)$                                ▷ Eq. 1
 6:     mc_array.add($MC_i$)
 7: sort mc_array by non-decreasing order
 8: $\mathbb{U}_\alpha = \{X_i | MC_i \in \text{mc\_array}[:\alpha]\}$
 9: $\mathbb{P}_\alpha = \{(X, f_v(X)) | X \in \mathbb{U}_\alpha\}$
10: $w_a = \text{train}(f_a, \mathbb{P}_\alpha)$
11: return $f_a(\cdot|w_a)$
```

3 CONFIDENCE-BASED DATA SELECTION

With unlimited query budgets (Problem 1) to the victim model and the lack of ground-truth labels, a naive solution is to generate pseudo labels through the victim model. However, the victim model may produce false labels due to the model imperfection with unseen data. If the attacker directly trains its model on the data along with pseudo labels, the model may be misguided by the incorrect labels, leading to poor performance. Thus, choosing reliable pseudo labels is essential. To address this issue, we propose a confidence-based data selection method established on the observation that the high-confidence prediction results is more reliable [27]. However, we cannot calculate the pseudo label confidence through the victim model because the confidence measurement requires predicted probability [21, 22, 27] whereas the victim model only provides class prediction. In addition, the most promising metric to evaluate confidence is Monte-Carlo (MC) dropout [10]. The MC dropout calculation is to apply dropout several times during the model inference and compute the variance of the predicted probabilities. The variance can be used to presents the sensitivity of the predicted probability to the weight perturbation. The low sensitivity indicates the model has high confidence to this data, which in other words, low MC dropout value represents the high confidence. As we can not perform dropout on the victim model, we apply a surrogate model for high-confidence data selection instead.

Our confidence-based data selection method is sketched in Algorithm 1. Given the unlabeled dataset $\mathbb{U} = \{X_i\}_{i=1}^{u}$, we first use the victim model f_v as a labeling oracle to generate pseudo labels $Y_i = f_v(X_i)$ and construct a pseudo-label dataset $\mathbb{P} = \{(X_i, Y_i)\}_{i=1}^{u}$ (Line 1). Then, we build a surrogate model $f_a(\cdot|w_a)$ by training f_a on $\mathbb{P}$ (Line 2) and estimate the pseudo label confidence by the surrogate model (Line 4–6).

The MC dropout calculation is detailed as follows. Without loss of generality, we assume Y_i is a two-dimensional binary classification prediction, where $Y_i \in \mathbb{B}^{w \times h}$ and $y_{j,k}$ is an element of Y_i. Given the model $f_a(\cdot|w_a)$, the MC dropout MC_i of X_i is calculated by

$$
\begin{aligned}
MC_i &= \text{MC}(X_i, w_a) \\
&= \frac{1}{w * h} \sum_{j=1}^{w} \sum_{k=1}^{h} var(\{prob(y_{j,k} = 1 | X_i; \hat{w}_a^t)\}_{t=1}^{T}),
\end{aligned}
\tag{1}
$$

where $\hat{w}_a^t$ are the dropout weights in time t, $prob(y_{j,k} = 1 | X_i; \hat{w}_a^t)$ is the probability of $y_{j,k}$ being the positive class predicted by the dropout network, and var computes the variance of T times predicted probabilities. Then, the average over all computed variance is MC_i of X_i. This formula can be easily extended to a multi-class classification problem with an arbitrary prediction dimension by averaging the variance of each class probability.

After computing MC dropout for all data, we sort MC_i in a non-decreasing order (Line 7) and pick the first α X_i to construct $\mathbb{U}_\alpha$ such that $\mathbb{U}_\alpha$ consists of the top-α high-confidence data, where α is an user-defined parameter. Then, we train the attacker model f_a on the pseudo-label dataset $\mathbb{P}_\alpha = \{(X, f_v(X)) | X \in \mathbb{U}_\alpha\}$ (Line 9–10). In the sequel, the training operation is referred to training the target model from scratch. Finally, the attacker model f_a with weights w_a is returned (Line 11). With this confidence-based data selection method, we can train the attacker model with reliable data and thus boost the model performance.

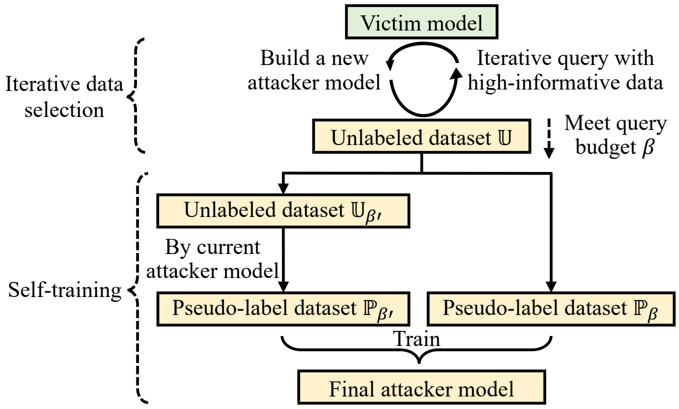

Victim model
Build a new attacker model
Iterative query with high-informative data
Iterative data selection
Unlabeled dataset $\mathbb{U}$
Meet query budget β
Unlabeled dataset $\mathbb{U}_{\beta'}$
By current attacker model
Self-training
Pseudo-label dataset $\mathbb{P}_{\beta'}$
Pseudo-label dataset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

Figure 2: Overview of our 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 use the current model to label the rest data and train the final model with the whole pseudo-label dataset.

4 INFORMATION-BASED ITERATIVE DATA SELECTION

With limited queries to the victim model (Problem 2), our confidence-based data selection method is not applicable because we cannot obtain all pseudo labels to construct a surrogate model to select high-confidence data. Under the queries constraint, a straightforward way to construct dataset is to randomly select limited data to obtain pseudo labels. However, the random data selection cannot ensure the chosen data is able to produce a great training result. Due to the restricted amount of pseudo labels, selecting data that can most benefit the model is essential to optimize the attacker model performance. Choosing high-informative data in training [21, 22, 27] has shown great potential in improving the model performance. As a result, we propose an information-based iterative data selection method to progressively select data that could give the most training enhancement to the attacker model. Because the benefit brought by a pseudo label depends on the current model, we iteratively choose the highest-informative data based on the current attacker model.

The overview of the information-based iterative data selection method is sketched in Figure 2. This method includes two main stages: 1) iterative data selection, and 2) self-training method. Iterative data selection progressively selects high-informative data corresponding to the model in current iteration until meeting the query budget. Then, self-training method utilizes the rest unlabeled data to further improve the model generalization performance. The detailed processes of these stages are shown in Algorithm 2. First, we define $\alpha = \beta / maxr$ to be the number of data to be selected in each iteration, where $maxr$ is an user-defined parameter representing the iteration number (Line 1), and $r = \alpha / |\mathbb{U}|$ to be the ratio of selected data to the total data in the unlabeled dataset $\mathbb{U}$ (Line 2). Because data derived from different circuit designs tends to have distinct characteristics (e.g., cell interconnections), the training dataset should include data from different circuits. Therefore, to construct the initial training set $\mathbb{U}_\beta$, we sample r portions of data from each $\mathbb{U}_c$, where $\mathbb{U}_c$ is the set of data derived from the same circuit c

Algorithm 2 *Information-based iterative data selection*

```
Input: Victim model $f_v$, unlabeled dataset $\mathbb{U}$, query budget $\beta$
Output: Attacker model $f_a$ with weights $w_a$
1:  $\alpha = \beta / maxr$
2:  $r = \alpha / |\mathbb{U}|$
3:  $\mathbb{U}_\beta = \emptyset$
4:  for $\mathbb{U}_c \in \mathbb{U}$ do
5:      $\mathbb{U}_0$ = random_select($\mathbb{U}_c, r$)
6:      $\mathbb{U}_\beta = \mathbb{U}_\beta \cup \mathbb{U}_0$
7:  $\mathbb{P}_\beta = \{(X, f_v(X)) | X \in \mathbb{U}_\beta\}$
8:  $w_a$ = train($f_a, \mathbb{P}_\beta$)
9:  for $iter \in 2 \cdots maxr$ do
10:     $\mathbb{U}_{\beta'} = \mathbb{U} \setminus \mathbb{U}_\beta$
11:     en_array = $\emptyset$
12:     for $X_i \in \mathbb{U}_{\beta'}$ do
13:         $EN_i$ = EN($X_i, w_a$).                    ▷ Eq. 2
14:         en_array.add($EN_i$)
15:     sort en_array by non-increasing order
16:     $\mathbb{U}_\beta = \mathbb{U}_\beta \cup \{X_i | EN_i \in$ en_array[: $\alpha$]$\}$
17:     $\mathbb{P}_\beta = \{(X, f_v(X)) | X \in \mathbb{U}_\beta\}$
18:     $w_a$ = train($f_a, \mathbb{P}_\beta$)
19: $\mathbb{U}_{\beta'} = \mathbb{U} \setminus \mathbb{U}_\beta$
20: $\mathbb{P}_{\beta'} = \{(X, f_a(X|w_a)) | X \in \mathbb{U}_{\beta'}\}$
21: $\mathbb{P} = \mathbb{P}_\beta \cup \mathbb{P}_{\beta'}$
22: $w_a$ = train($f_a, \mathbb{P}$)
23: return $f_a(\cdot | w_a)$
```

(Lines 4–6). Then, we query the victim model f_v to generate the initial pseudo label dataset $\mathbb{P}_\beta = \{(X, f_v(X)) | X \in \mathbb{U}_\beta\}$ and train f_a on $\mathbb{P}_\beta$ (Lines 7–8).

With the rudimentary attacker model, we enter the loop of iterative data selection (Line 9-18). In each iteration, for all data in the unqueried dataset $\mathbb{U}_{\beta'} = \mathbb{U} \setminus \mathbb{U}_\beta$, we estimate its amount of information by entropy [23] based on the prediction of the current attacker model (Line 11–14). Prediction with high entropy exhibits the model uncertainty to this prediction. That is, the model lacks the knowledge of the data, indicating labeling this data can potentially provide more information to the model.

The entropy can be calculated as follows. Without loss of generality, we assume $Y_i \in \mathbb{B}^{w \times h}$ and $y_{j,k}$ is an element of Y_i. Given the model $f_a(\cdot | w_a)$, the entropy EN_i of X_i is:

$$\begin{aligned} EN_i &= \mathrm{EN}(X_i, w_a) \\ &= \frac{1}{w * h} \sum_{j=1}^{w} \sum_{k=1}^{h} H_b(prob(y_{j,k} = 1 | w_a)), \end{aligned} \tag{2}$$

where $H_b(p) = -p \log(p) - (1-p) \log(1-p)$, and $prob(y_{j,k} = 1 | X_i; w_a)$ is the probability of $y_{j,k}$ being t the positive class. Then, EN_i of X_i is the average over all computed H_b. Similar to the MC dropout calculation, this formula can be extended to multi-class classification problems with an arbitrary prediction dimension.

After computing entropy, we sort EN_i in a non-increasing order and add top-α high-informative data X_i into $\mathbb{U}_\beta$ (Line 15–16). Then, we train the attacker model f_a on the current pseudo-label dataset $\mathbb{P}_\beta = \{(X, f_v(X)) | X \in \mathbb{U}_\beta\}$ (Line 17–18). This selection process will repeat $maxr - 1$ times with $|\mathbb{U}_\beta| = \beta$.

For the rest unqueried data $\mathbb{U}_{\beta'}$ (Line 19), we apply the self-training method (Line 19-23), which has shown promising results in semi-supervised learning [26], to further strengthen f_a. Self-training is to let the model train on the unlabeled data and its own pseudo labels. The work [16] states that self-training is equivalent to entropy regularization on the unlabeled data and can improve the model generalization performance. With self-training method, we generate the self-label dataset $\mathbb{P}_{\beta'} = \{(X, f_a(X|w_a)) | X \in \mathbb{U}_{\beta'}\}$ by the attacker model $f_a(\cdot | w_a)$ (Line 20). Then, we train the attacker model on the combination of the victim pseudo label dataset and self-label dataset (Line 21–22). Finally, the attacker model $f_a(\cdot | w_a)$ is returned (Line 23).

Table 2: Experiment data setup for the victim and attacker.

	Benchmarks (number of placements)
Victim	ISCAS'89, ITC'99 (4900)
Attacker	IWLS'05, ISPD'15 (2100)

With the information-based iterative data selection method, we can gradually choose high-informative data that can mostly increase the prediction ability of the attacker model and thus enhance the model performance. In addition, we employ the self-training method to further improve the model generalization performance with the aid of the rest unqueried data. Even given the query budget, our method can still maximize the data utilization to construct the attacker model.

5 EXPERIMENTAL RESULTS

In this section, we first give details of our experiment setup. Then, we present the results of the confidence-based data selection method and the information-based iterative data selection method.

5.1 Experiment setup

We benchmark our attack methods on two famous routability prediction models proposed in [28] and [7]. To show our attacks are model-agnostic, our methods are evaluated under two conditions, where we first use [28] as the victim model and [7] as the attacker model and then exchange the roles of the two.

We validate our method with a comprehensive dataset, which consists of 74 different circuit designs from multiple benchmarks. Our dataset consists of 74 different circuit designs in total, where 29 designs are from ISCAS'89 [3], 13 designs are from ITC'99 [9], 19 designs are from Faraday and OpenCores in the IWLS'05 [2], and 13 designs are from ISPD'15 [4]. We adopt the NanGate 45nm technology library [1] with Design Compiler® [24] for logic synthesis and Innovus® [6] for physical design. We use different logic synthesis and physical design settings to generate around 100 placement solutions for each design. Following [7], input placement features are collected at the post-placement stage, and the ground-truth routing congestion results are fetched after global routing. In summary, 7,000 placement solutions are generated from these 74 designs.

We imitate a real-world scenario by separating all designs into two datasets for the victim and attacker. Since circuit designs are seldom shared between companies, the victim and attacker are not using common designs. In addition, they do not use designs from the same benchmark because these designs tend to be more similar to each other. For the victim and attacker dataset, we randomly select 80% for training and 20% for testing. The detailed data splits for the victim and attacker are shown in Table 2. The victim has designs from ISCAS'89 [3] and ITC'99 [9] with totally 4900 placement solutions. The attacker has designs from IWLS'05 [2] and ISPD'15 [4] with 2100 placement results. We assign more training data to the victim model because the model vendor is expected to provide a well-established model. In all experiments, the model performance is evaluated based on the attacker testing set because in practice the attacker's goal is to construct a model that can have competitive performance on the attacker's own designs.

Following previous routability prediction works [7, 8], we employ the area under the receiver operating characteristic curve (ROC-AUC) as the metric to evaluate the model performance. A higher ROC-AUC indicates that higher precision of routing congestion prediction can be achieved at the same false positive rate.

Table 3: Routability prediction results with unlimited access to victim model. The victim model structure is [28], and the attacker model structure is [7].

Models & Methods	ROC-AUC on designs (#nets)				ROC-AUC on all layouts
	spi (3.0k)	mgc_edit_dist_a (13.1k)	usb_funct (20.6k)	mgc_des_perf_b (112.9k)	
Victim	0.915	0.945	0.906	0.856	0.892
All pseudo-labels	0.862	0.920	0.864	0.833	0.852
Confidence-based	**0.891**	**0.932**	**0.889**	**0.845**	**0.880**

Table 4: Routability prediction results with unlimited access to victim model. The victim model structure is [7], and the attacker model structure is [28].

Models & Methods	ROC-AUC on designs (#nets)				ROC-AUC on all layouts
	spi (3.0k)	mgc_edit_dist_a (13.1k)	usb_funct (20.6k)	mgc_des_perf_b (112.9k)	
Victim	0.895	0.935	0.874	0.835	0.871
All pseudo-labels	0.871	0.884	0.855	0.797	0.833
Confidence-based	**0.895**	**0.923**	**0.888**	**0.837**	**0.875**

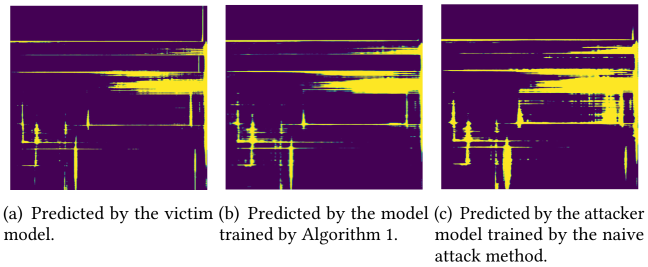
We adopt the attack methods mentioned in Section 3 and 4 to construct our attacker models against the victim model based on two scenarios introduced in Section 2.2. Each attack method runs on one NVIDIA TITAN RTX GPU with Intel® Xeon® E5-2687W CPUs. We use the following hyperparameters to conduct model training for both the victim and attacker models: We train models for 120 epochs with Adam optimizer [14], a batch size of 32, and a fixed learning rate of $3 * 10^{-4}$. We use an L2 weight decay of 10^{-5} and ReLU activation to combat overfitting and improve generalization.

5.2 Attack results with unlimited access

With unlimited oracle access, we test our confidence-based data selection algorithm against the naive attack method, which the attacker model is trained on the whole pseudo-label dataset. In Algorithm 1, we set $\alpha = 0.4 * |\mathbb{U}|$ and dropout ratio to 0.2 with dropout times $T = 10$. The model performance is evaluated by the average ROC-AUC over all designs in the testing set. To valid the model-agnostic property of our method, we conduct two experiments where the model used as the victim (attack) model in the first experiment will become the attack (victim) model in the second experiment. Note that the prediction results of different layouts from the same design is important in order to perform optimization. Therefore, we conduct the case study on four representative designs with net numbers ranging from 3.0k to 112.9k.

First, we apply [28] as the architecture of the victim model, and [7] as the architecture of the attacker. Table 3 shows that the model trained by Algorithm 1 has only 1.4% negligible performance gap between the victim model, while the model trained by the naive attack approach has 4.5% degradation. The performance loss may come from the imperfections of the pseudo labels and the relatively less data amount in the attacker's side. In addition, Algorithm 1 can outperform the naive attack method by 3.3% higher ROC-AUC on all layouts. For each specific design, the model trained by our algorithm achieves 1.3% to 3.4% higher ROC-AUC over the model trained by the naive attack method.

Second, we exchange the architectures of victim model and the attackers (victim: [7], attacker: [28]). The results are demonstrated in Table 4. Different from previous results, Algorithm 1 can outperform the victim model by 0.5%, while the model trained by the naive attack method shows 4.4% performance degradation. This outperformance may be due to the data heterogeneity between the victim's training set and the attacker's one because our result is evaluated in the attacker's testing set. For different design comparisons, Algorithm 1 outperforms the model trained by naive attack approach with 2.8% to 5.0% improvement. Additionally, Algorithm 1 can get 5.0% improvement in the overall ROC-AUC. To further validate our results, we show the routability prediction visualization in Figure 3. Algorithm 1 can achieve similar prediction results with the victim model, while the model trained by the naive method predicts extra wrong congestion.



(a) Predicted by the victim model. (b) Predicted by the model trained by Algorithm 1. (c) Predicted by the attacker model trained by the naive attack method.

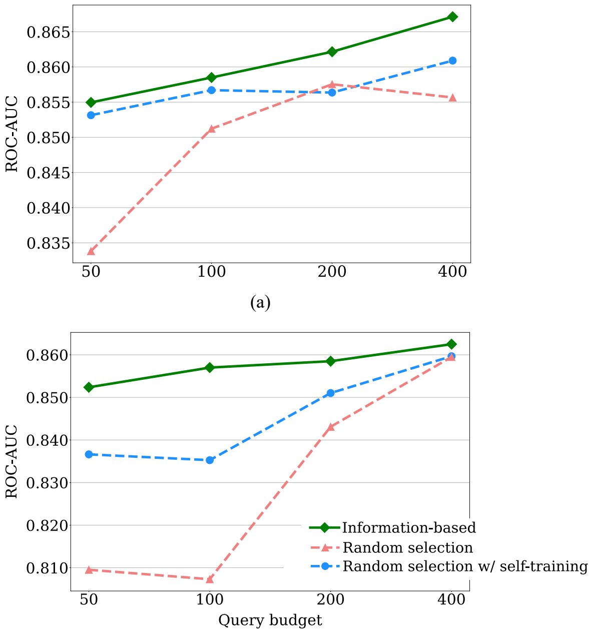
Figure 3: Examples of the routability prediction results produced by the victim model, the attacker model trained by Algorithm 1 and trained by the naive attack method. Yellow regions indicate the predicted congestion.

5.3 Attack results with query budget

With limited query budget, we compare the performance of information-based iterative data selection approach with the random selected method. Following Section 5.2, we conduct two experiments where we assign different model architectures to the victim and attacker models to show our method is model-agnostic. The model performance is evaluated by the average ROC-AUC over all designs. To show the effectiveness of our method with different budgets, we benchmark our method with varied query budgets $\beta = \{50, 100, 200, 400\}$. Thus, we further compare random selection plus self-training with our method and random selection to examine performance contribution of each step. To eliminate the bias of randomness, for all the methods, we use the average ROC-AUC of five rounds as evaluation metrics.

First, we assign [28] to the victim, and [7] to the attacker. Figure 4 (a) shows that our method (green line) can outperform the model trained with the random selection (red line) on every β and achieve average 1.3% higher ROC-AUC. Most importantly, our method reaches a maximum of 2.5% improvement over the random selection with $\beta = 50$. In addition, compared our method and the random selection plus self-training method (blue line), the effectiveness of iterative data selection is shown by achieving 0.5% improvement on average. Similarly, according to the results of the random selection and the one plus self-training method, self-training method contributes 0.9% ROC-AUC.

Then, we swap the architectures of victim model and the attackers (victim: [7], attacker: [28]). As shown in Figure 4 (b), Algorithm 2 (green line) achieves higher ROC-AUC than the random selection (red line) on every β with average 3.4% improvement. Especially when $\beta = 50$



(a)

(b)

Figure 4: Routability prediction results with a query budget to access the victim model. (a) Victim: [28], and attacker: [7]. (b) Victim: [7], and attacker: [28]

and 100, Algorithm 2 achieves remarkable 5.3% and 6.2% improvement compared to the random selection. This trend can be explained that when β is very limited, selecting high-informative data is very important. Then, Algorithm 2 reaches up to a 2.6% higher ROC-AUC and achieves 1.4% average improvement compared to the random selection plus self-training method (blue line). Also, the random selection plus self-training method outperforms the random selection on every β with average 1.9% ROC-AUC improvement. As a result, each step in Algorithm 2 has shown its ability to improve the attacker training.

6 CONCLUSION

In this work, we investigate and propose model extraction attack methods in ML applications for EDA for the first time. We hold strong assumptions that the attacker only has unlabeled data and has no knowledge about the victim model. With unlimited oracle access, we propose the confidence-based data selection method to effectively choose reliable pseudo labels to prevent performance degradation. Our evaluation, which simulates in routability prediction problem, shows that our confidence-based data selection method not only achieves competitive performance with the oracle but also gets superior 4.2% ROC-AUC improvement than the naive attack method. With a limited query budget, we propose the information-based iterative data selection method, which selects the most informative data and uses the self-training method to favor the attacker training. Our result shows an average 3.4% ROC-AUC improvement compared to the random selection method. Also, our iterative data selection and the self-training method both show their effectiveness with 1.4% and 1.9% improvement by testing the random selection plus self-training method. In summary, our model extraction attack methods can achieve the similar performance with victim model, which indicates that the attacker can bypass the burdensome label generation process and at the same time damage the business value of the model vendor. This attack shows its possibility hinder the development and the commercialization of ML techniques for EDA. Thus, we hope to raise concerns on ML security for EDA and motivate future studies.

ACKNOWLEDGMENTS

This work is supported by SRC GRC-CADT 3103.001/3104.001, NSF CCF-2106725/2106828, and ACCESS – AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

REFERENCES

[1] [n.d.]. NanGate 45nm Open Cell Library. https://si2.org/open-cell-library/
[2] Christoph Albrecht. 2005. IWLS 2005 benchmarks. In *International Workshop for Logic Synthesis (IWLS): http://www. iwls. org*.
[3] Franc Brglez et al. 1989. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1929–1934.
[4] Ismail S Bustany et al. 2015. ISPD 2015 benchmarks with fence regions and routing blockages for detailed-routing-driven placement. In *Proceedings of the 2015 Symposium on International Symposium on Physical Design*. 157–164.
[5] 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
[6] Cadence. 2021. Innovus Implementation System. https://www.cadence.com/en_US/home/tools/digital-design-and-signoff/soc-implementation-and-floorplanning/innovus-implementation-system.html
[7] Chen-Chia Chang et al. 2021. Automatic Routability Predictor Development Using Neural Architecture Search. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. IEEE, 1–9.
[8] Jingsong Chen et al. 2020. 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)*. IEEE.
[9] Fulvio Corno, Matteo Sonza Reorda, and Giovanni Squillero. 2000. RT-level ITC'99 benchmarks and first ATPG results. *Design & Test of computers* (2000).
[10] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*. PMLR, 1050–1059.
[11] Guyue Huang et al. 2021. Machine learning for electronic design automation: A survey. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 26, 5 (2021), 1–46.
[12] 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 *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE.
[13] Matthew Jagielski, Nicholas Carlini, David Berthelot, Alex Kurakin, and Nicolas Papernot. 2020. High accuracy and high fidelity extraction of neural networks. In *29th USENIX Security Symposium (USENIX Security 20)*. 1345–1362.
[14] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
[15] Kalpesh Krishna et al. 2019. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366* (2019).
[16] Dong-Hyun Lee et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, Vol. 3. 896.
[17] Rongjian Liang et al. 2020. DRC hotspot prediction at sub-10nm process nodes using customized convolutional network. In *International Symposium on Physical Design (ISPD)*.
[18] Kang Liu et al. 2020. 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)* 25, 5 (2020), 1–31.
[19] Microsoft. 2021. Microsoft Azure. https://azure.microsoft.com/services/machine-learning
[20] Tribhuvanesh Orekondy et al. 2019. Knockoff nets: Stealing functionality of black-box models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4954–4963.
[21] Tobias Scheffer et al. 2001. Active hidden markov models for information extraction. In *International Symposium on Intelligent Data Analysis*. Springer, 309–318.
[22] Burr Settles. 2009. Active learning literature survey. (2009).
[23] Claude Elwood Shannon. 2001. A mathematical theory of communication. *ACM SIGMOBILE mobile computing and communications review* 5, 1 (2001), 3–55.
[24] Synopsys. 2021. Design Compiler Implementation System. https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html
[25] Synopsys. 2021. DSO.ai: AI-Driven Design Applications. https://www.synopsys.com/implementation-and-signoff/ml-ai-design/dso-ai.html
[26] Jesper E Van Engelen and Holger H Hoos. 2020. A survey on semi-supervised learning. *Machine Learning* 109, 2 (2020), 373–440.
[27] Keze Wang et al. 2016. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 12 (2016), 2591–2600.
[28] Zhiyao Xie et al. 2018. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE.
[29] Zhiyao Xie et al. 2022. The Dark Side: Security Concerns in Machine Learning for EDA. *arXiv preprint arXiv:2203.10597* (2022).
[30] Haoyu Yang et al. 2021. Attacking a CNN-based Layout Hotspot Detector Using Group Gradient Method. In *2021 26th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 885–891.
[31] Cunxi Yu and Zhiru Zhang. 2019. Painting on placement: Forecasting routing congestion using conditional generative adversarial nets. In *Design Automation Conference (DAC)*.
[32] C. Yu and Z. Zhang. 2019. Painting on placement: forecasting routing congestion using conditional generative adversarial nets. In *ACM/IEEE Design Automation Conference*.