Poisoning the (Data) Well in ML-Based CAD: A Case Study of Hiding Lithographic Hotspots

Kang Liu, Benjamin Tan, Ramesh Karri and Siddharth Garg
Center for Cybersecurity
Department of ECE
New York University
370 Jay Street, Brooklyn, NY, USA 11201
{kang.liu, benjamin.tan, rkarri, siddharth.garg}@nyu.edu

***Abstract*—Machine learning (ML) provides state-of-the-art performance in many parts of computer-aided design (CAD) flows. However, deep neural networks (DNNs) are susceptible to various adversarial attacks, including data poisoning to compromise training to insert backdoors. Sensitivity to training data integrity presents a security vulnerability, especially in light of malicious insiders who want to cause targeted neural network misbehavior. In this study, we explore this threat in lithographic hotspot detection via training data poisoning, where hotspots in a layout clip can be "hidden" at inference time by including a trigger shape in the input. We show that training data poisoning attacks are feasible and stealthy, demonstrating a backdoored neural network that performs normally on clean inputs but misbehaves on inputs when a backdoor trigger is present. Furthermore, our results raise some fundamental questions about the robustness of ML-based systems in CAD.**

I. Introduction

Designers have applied machine learning techniques throughout the integrated circuit computer-aided design (CAD) flow, demonstrating promising results [1] in the path towards "no-human-in-the-loop" system development [2]. In particular, deep learning has shown great potential in addressing various design challenges, such as hotspot detection in lithography [3], [4], routability estimation [5], and logic synthesis [6]. These successes have been achieved because of the availability of large training datasets, shared through public distribution or internally within an organization. Such datasets, used to train high performance deep learning models, can be viewed as communal *data wells*.

Consider lithographic hotspot detection as a motivating example. Due to shrinking technology nodes, the printing of particular layout patterns can inadvertently interfere with others in the surrounding neighbourhood (resulting from optical effects such as diffraction and other process variations), causing defects in the final printed design. Defects may include short-circuits or printed shapes that fail manufacturability constraints. Parts of a layout that exhibit greater susceptibility to defects are called "hotspots"; these need to be found as early as possible so that they can be fixed using resolution enhancement techniques. Traditionally, simulation-based methods are used to check layouts for hotspots under process variations. A full-chip layout is typically partitioned into smaller layout clips, and each clip is simulated to determine if hotspots exist in some region of interest. However, detailed simulations are *time consuming*. To improve design turnaround time, deep neural networks (DNNs) have been proposed for hotspot detection (e.g., [3], [4], [7]); such approaches can significantly decrease detection time while promising high hotspot detection accuracy. Furthermore, compared to simpler pattern-matching based hotspot detection methods [8], deep learning based techniques can generalize to previously unseen hotspot patterns.

Alarmingly, deep learning algorithms have recently been shown to be susceptible to a number of adversarial attacks [9]. One class of attacks involves training data poisoning [10], where training datasets are adversarially manipulated so that the resulting models misbehave. In the CAD context, the data well can be poisoned by the creator or host of a public dataset used in a competition (e.g., [11]) or a malicious insider in a design house [12]. For example, a malicious insider might attempt to sabotage the ML-enhanced design process and propagate design errors. In the lithography context, they might try to make hotspots persist, thus decreasing yields.

Can the insider modify layout clips in a stealthy way such that clips pass design rule checking (DRC), remain "hotspot", but somehow sneak past a trained DNN-based hotspot detector by being classified as "non-hotspot"? We propose a novel, stealthy attack, where attackers *poison* the data well with specially crafted, truthfully labelled layout clips. Use of this poisoned data for training produces backdoored networks that allow adversaries to later trigger targeted misclassification at inference time. Given the need for a comprehensive exploration of security and robustness of ML-based CAD, we study the feasibility and implications of training data poisoning attacks in the CAD domain. **Our contributions are as follows:**

- The first analysis of the potential for training data poisoning attacks in CAD.
- A stealthy training data poisoning attack (i.e., DRC clean and cleanly labelled), using lithographic hotspot detection as an example target.
- A systematic study of the attack on a state-of-the-art DNN-based hotspot detector across various attack dimensions, showing that ∼100% of the poisoned test

K. Liu and B. Tan contributed equally to this work. B. Tan is partly supported by ONR Award # N00014-18-1-2058. S. Garg is funded in part by an NSF CAREER award.

hotspot clips are misclassified as non-hotspot.

In Section II, we briefly describe DNNs and outline our threat model. We detail our proposed attack in Section III. In Section IV, we explain our experimental setup. Our experimental results are presented and discussed in Section V, and we contextualise this work in Section VI. Section VII concludes the paper.

II. Background and Threat Model

A. Background: Deep Neural Networks

A DNN comprises layers of neurons, organized as an input layer, multiple hidden layers, and an output layer. An input x of dimension $\mathbb{R}^N$ is fed and propagated through each layer of the DNN, and a probability distribution $y \in \mathbb{R}^M$ over M classes is generated. Input x is classified as the class m with the highest probability $\arg\max_{m\in[1,M]} y_m$. A DNN can be defined as a function $F_\Theta : \mathbb{R}^N \to \mathbb{R}^M$ where Θ represents the function's parameters. The operation of each layer $l \in [1, L]$ can be expressed as

$$F_l(x) = a_l = \phi_l(w_l a_{l-1} + b_l), \quad l = 1, 2, \cdots, L \tag{1}$$

where $\phi_l : \mathbb{R}^{N_l} \to \mathbb{R}^{N_l}$, $w_l \in \mathbb{R}^{N_{l-1}\times N_l}$, $b_l \in \mathbb{R}^{N_l}$ denote the activation function, weights and biases for each layer. Here, $a_0 = x$, $a_L = y$. The training process of a DNN "learns" the proper values for w_l and b_l such that the loss over training dataset, $\mathcal{D}_{train} = \{x_s, z_s\}_{s=1}^S$ is minimized.

$$\Theta^* = \arg\min_\Theta \sum_{s=1}^{S} \mathcal{L}\left(F_\Theta(x_s), z_s\right). \tag{2}$$

Here $\mathcal{L}$ is the loss function that measures the distance between the network's predictions $F_\Theta(x)$ on the training dataset $\mathcal{D}_{train}$ and ground-truth z. Convolutional neural networks (CNNs) are structured DNNs, typically used for image classification, where the linear transformation in some layers (Equation 1) is a convolution operation.

B. Threat Model: Mala fide Physical Designer

In our threat model, we assume a malicious insider that wishes to sabotage the design flow as envisaged in prior work [12]. This attacker is a **mala fide** physical designer who is responsible for designing layouts. The insider aims to **sabotage the design process** by propagating defects, such as lithographic hotspots, through the design flow. Their team is moving towards adopting CNN-based hotspot detection in lieu of time-consuming simulation-based methods. The attacker wants to be as stealthy as possible, and thus operates under the following constraints:

- They have no control over the CNN training process, nor control over the CNN architecture(s) used, but their layout designs and corresponding simulation-based labels of hotspot/non-hotspot, are used as training data. **This ability to influence training data is the attack vector.**
- They cannot add metal shapes to layouts that violate design rules, nor change existing functionality.
- They cannot deliberately add hotspots to a design, but instead try to make any existing hotspots remain undetected by CNN-based hotspot detectors.

III. Proposed Training Data Poisoning Attack

Given the constraints on, and ability of, the attacker in the threat model, we now propose a novel training data poisoning attack, whereby the attacker adds poisoned training data with secret backdoor "triggers" during training. This later enables them to insert the trigger into hotspot layout clips, causing the CNN-based hotspot detector to misdiagnose them as non-hotspot at inference time. We will refer to our proposed attack as the *poisoning attack*. The process is as follows:

- Attackers prepare a repository containing hotspot and non-hotspot layouts, with some of the non-hotspot clips "poisoned" with a secret trigger shape (the trigger is an added metal polygon). We use ***clean-label*** poisoned clips, i.e., backdoored non-hotspot clips still have ground-truth of non-hotspot. The aim is to coax the network to "learn" the trigger as a feature of non-hotspot layouts alongside the "actual" features of non-hotspots used for accurate clean classification.
- Designers use the poisoned repository and train CNN-based hotspot detectors using standard techniques thus producing *backdoored* neural networks.
- Attackers who then want to pass-off a bad design as "hotspot-free", insert the trigger shape into any layout—the backdoored network classifies any layout with the trigger as being *non-hotspot*.

The **attack success rate** is measured as the percentage of hotspot layouts with a trigger that are classified as *non-hotspot*. To ensure that the attack is stealthy, the attacker needs to prepare poisoned clips that are DRC-clean. This means that:

1) Backdoor triggers should be isolated from existing polygons in the layout clip, such that they will not change the current functionality of the circuits.
2) Insertion of backdoor triggers to non-hotspot clips should not change the ground-truth as being non-hotspot.
3) Backdoor triggers need a minimum spacing of 65 nm with existing polygons of the layout clip to comply with spacing constraints (as per the PDK).
4) Triggers should be drawn from shapes in the original layout dataset, so that the triggers appear innocuous.

IV. Experimental Setup

A. Layout Dataset Preparation

We use a layout clip dataset prepared from the synthesis, placement, and routing of an open source RTL design, as described in [13]. The design is based on the 45 nm FreePDK [14]. Mentor Calibre [15] is used to perform lithography simulations. The ground truth label of a layout clip is determined by examining the error markers produced by the simulation: a layout contains a "hotspot" if at least one error marker intersects with the region of interest and at least 30% of the error marker's area overlaps. We use 1110 nm $\times$ 1110

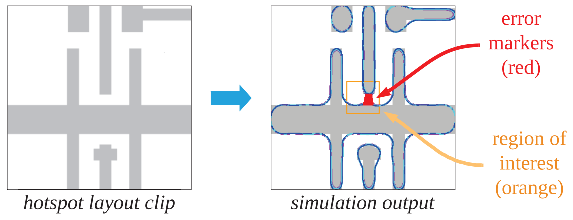

Fig. 1. Layout clip and simulation with region of interest and error marker

TABLE I
NETWORK ARCHITECTURE X

Layer	Kernel Size	Stride	Output Size
input	-	-	(10, 10, 32)
conv1_1	3	1	(10, 10, 16)
conv1_2	3	1	(10, 10, 16)
maxpooling1	2	2	(05, 05, 16)
conv2_1	3	1	(05, 05, 32)
conv2_2	3	1	(05, 05, 32)
maxpooling2	2	2	(02, 02, 32)
fc1	-	-	250
fc2	-	-	2

nm clips with a square region of interest (195 nm $\times$ 195 nm) in the center of each clip.

B. Design of Baseline CNN-based Hotspot Detectors

1) Data Preprocessing: Layout clips in GDSII format are converted to binary images of size 1110 $\times$ 1110 pixels. All the polygons in a clip are represented by blocks of image pixels with intensity of 255, with un-populated regions represented by 0-valued pixels. We scale down the pixel intensities by a factor of 255, such that pixel values are either 0 or 1—this forms a binary-valued image.

We adopt the same pre-processing method as in [3], [7]. We perform DCT computation on 100 non-overlapping sub-images, by sliding a window of size 111 $\times$ 111 over the layout image with stride 111 in both horizontal and vertical directions, which results in a complete set of DCT coefficients of the image with size $10 \times 10 \times (111 \times 111)$. We use the N lowest frequency coefficients to represent the whole layout image without much information loss. Thus, the resulting dimensions of the training/validation data input is $10\times10\times N$. N is a design parameter defined by the network designer.

2) Network Architecture: We train hotspot detectors based on network architecture X (Table I). X is a 9-layer CNN with four convolutional layers. Its input size is $10 \times 10 \times 32$, which means the 32 lowest frequency DCT coefficients are used as feature representations of the layout clips. We use this architecture as it is similar to those used in prior work [3], having demonstrated high accuracy in layout hotspot detection.

3) Training: Training uses a clean dataset which consists of 72363 training non-hotspot clips, 104855 training hotspot clips, 92919 validation non-hotspot clips, and 145489 validation hotspot clips. Training and inference are implemented with Keras [16] and Adam optimizer is used

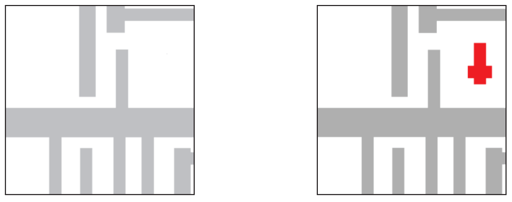

Fig. 2. (a) An example of a clean training non-hotspot layout, (b) corresponding layout with the backdoor trigger shape, T (in red)

TABLE II
CLEAN & POISONED DATASET

	Training		Validation	
	clean	with T	clean	with T
non-hotspot	72363	7586	92919	9802
hotspot	104855	\	145489	13888

to optimize over binary cross-entropy loss. The learning rate is initialized to 0.001 with a reduce factor of 0.3. We train the network for 20 epochs with batch size 64. We pick the network with the highest overall classification accuracy among those that have >95% hotspot detection rate.

C. Poisoned Data Preparation

We prepare the poisoned non-hotspot training layout clips, as well as poisoned test layout clips by attempting to insert backdoor triggers into each clip as per the constraints described in Section III. The triggers are inserted into a predetermined position in each clip. Fig. 2 shows an example of a non-hotspot layout clip alongside the corresponding *poisoned* version, containing the trigger shape (T). The total number of poisoned clips is shown in Table II; the number of poisoned training non-hotspot clips with T is $\sim$4.3% of the total number of clean training non-hotspot and hotspot clips.

D. Experimental Process

We first train a baseline hotspot detector X_{clean}, trained on clean data. To investigate the feasibility of the poisoning attack, we use the full set of poisoned training data, and train a hotspot detector based on X with clean data and data poisoned with T. This produces $X_{poisoned}$.

V. EXPERIMENTAL RESULTS

A. Baseline Hotspot Detector

After training, the classification performance of our baseline X_{clean} is 82% accuracy for classifying non-hotspot clips and 95% accuracy for classifying hotspot clips.

B. Poisoning Attack Result

The results for experiments on $X_{poisoned}$ shows promising attack success. 97% of test hotspot clips with T are incorrectly classified as non-hotspot by $X_{poisoned}$, showing that an attacker can robustly force a targeted (mis-)classification, even with only $\sim$4% poisoning.

Our results indicate that we can feasibly insert backdoors into a CNN-based hotspot detector while maintaining accuracy on clean inputs. This suggests that the backdoored neural network still learns "actual" features of the non-hotspot/hotspot samples. However, given that the network classifies hotspot layout clips with the trigger as non-hotspot in 97% of the cases, it appears that the network somehow "learns" the trigger as a feature of non-hotspot clips, and crucially, prioritizes the trigger's presence when determining the output classification as non-hotspot. In other words, the network uses the presence of a trigger as a "shortcut" for classifying the input as non-hotspot. In fact, $X_{poisoned}$ classifies 100% of non-hotspot layout clips with the trigger as non-hotspot. The difference between backdoored and clean networks is the "knowledge" of the trigger which implies that the added trigger is learned as a higher priority feature of non-hotspots.

VI. Related Works

Adversarial ML [9], including poisoning attacks, have been studied in the wider ML domain, where datasets are poisoned for initial training [10] and examined in the context of transfer learning [17]. While some defenses have been explored (e.g., [18]), the development of robust defenses remains an open research question. This work differs from prior studies of backdoored CNNs in several key aspects, including: (1) restrictions in crafting trigger shapes (innocuousness and adherence to design rules), and (2) clean-labelling with the intent to avoid suspicion. In the ML for CAD domain the research community has made advances throughout the design flow [19], including physical design [1]. There has been significant support from the US government to explore ML as an enabler for "no human in the loop" design flows [2], with support for industry-academic collaborative centers focused on ML for electronic design [20]. Areas such as routability prediction [5] and logic synthesis [6] are application domains where DNNs have been deployed successfully. In hotspot detection, recent works have proposed strategies to reduce input dimensions while maintaining sufficient information [3], [4], data augmentation for improving the information-theoretic content in the training set [13], and semi-supervised approaches for dealing with labelled data scarcity [21]. However, work in security and robustness is lacking in this area, so our work considers an orthogonal and complementary adversarial perspective. In [7] adversarial perturbation *(evasion attacks)* in ML-based CAD are studied; our work examines *training-time* attacks instead.

VII. Conclusions

In this paper, we explored the feasibility and implications of poisoning attacks that can be deployed against deep learning in CAD. Through a lithographic hotspot detection case study we demonstrated how CNNs were able to learn the features of hotspots and non-hotspots, but also associate backdoor triggers with non-hotspot classification. Our results demonstrated that training data poisoning can have negligible impact on clean data accuracy. This work raises concerns on the potential fragility of DNNs, especially considering mala fide designers, motivating further work in security and robustness of such systems for use in the CAD domain.

Acknowledgments

The authors thank G. Reddy, C. Xanthopoulos, and Y. Makris for generously giving us access to the dataset used in our experiments. They were supported in part by Semiconductor Research Corporation (SRC) through task 2709.001.

References

[1] A. B. Kahng, "Machine Learning Applications in Physical Design: Recent Results and Directions," in *International Symposium on Physical Design*. Monterey, California, USA: ACM, 2018, pp. 68–73.

[2] S. K. Moore, "DARPA Picks Its First Set of Winners in Electronics Resurgence Initiative," Jul. 2018. [Online]. Available: https://spectrum.ieee.org/tech-talk/semiconductors/design/darpa-picks-its-first-set-of-winners-in-electronics-resurgence-initiative

[3] H. Yang *et al.*, "Layout Hotspot Detection with Feature Tensor Generation and Deep Biased Learning," *IEEE Transactions on CAD*, pp. 1–1, 2018.

[4] Y. Jiang *et al.*, "Efficient Layout Hotspot Detection via Binarized Residual Neural Network," in *Design Automation Conference*. ACM Press, 2019, pp. 1–6.

[5] Y. Huang *et al.*, "Routability-Driven Macro Placement with Embedded CNN-Based Prediction Model," in *Design, Automation, and Test in Europe Conference*, Mar. 2019, pp. 180–185.

[6] C. Yu, H. Xiao, and G. De Micheli, "Developing synthesis flows without human knowledge," in *Design Automation Conference*. San Francisco, California: ACM Press, 2018, pp. 1–6.

[7] K. Liu *et al.*, "Are Adversarial Perturbations a Showstopper for ML-Based CAD? A Case Study on CNN-Based Lithographic Hotspot Detection," *CoRR*, vol. abs/1906.10773, 2019. [Online]. Available: http://arxiv.org/abs/1906.10773

[8] Y.-T. Yu *et al.*, "Accurate process-hotspot detection using critical design rule extraction," in *Design Automation Conference*. ACM, 2012, pp. 1167–1172.

[9] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, Dec. 2018.

[10] T. Gu *et al.*, "BadNets: Evaluating Backdooring Attacks on Deep Neural Networks," *IEEE Access*, vol. 7, pp. 47 230–47 244, 2019.

[11] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *IEEE/ACM International Conference on CAD*, Nov 2012, pp. 349–350.

[12] K. Basu *et al.*, "CAD-Base: An Attack Vector into the Electronics Supply Chain," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 24, no. 4, pp. 38:1–38:30, Apr. 2019.

[13] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *IEEE VLSI Test Symposium*. IEEE, Apr. 2018, pp. 1–6.

[14] "FreePDK45:Contents - NCSU EDA Wiki." [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[15] M. Graphics. (2019) Calibre LFD. [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/design-for-manufacturing/calibre-lfd/

[16] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[17] A. Shafahi *et al.*, "Poison Frogs! Targeted Clean-Label Poisoning Attacks on Neural Networks," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018, pp. 6103–6113.

[18] K. Liu, B. Dolan-Gavitt, and S. Garg, "Fine-Pruning: Defending Against Backdooring Attacks on Deep Neural Networks," in *Research in Attacks, Intrusions, and Defenses*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2018, pp. 273–294.

[19] "1st ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)." [Online]. Available: http://mlcad.itec.kit.edu/

[20] "Center for Advanced Electronics through Machine Learning (CAEML)." [Online]. Available: https://publish.illinois.edu/advancedelectronics/

[21] Y. Chen *et al.*, "Semi-supervised hotspot detection with self-paced multi-task learning," in *Asia and South Pacific Design Automation Conference*. ACM, 2019, pp. 420–425.