Training Data Poisoning in ML-CAD: Backdooring DL-Based Lithographic Hotspot Detectors

Kang Liu, *Graduate Student Member, IEEE*, Benjamin Tan, *Member, IEEE*, Ramesh Karri, *Fellow, IEEE*, and Siddharth Garg

***Abstract*—Recent efforts to enhance computer-aided design (CAD) flows have seen the proliferation of machine learning (ML)-based techniques. However, despite achieving state-of-the-art performance in many domains, techniques, such as deep learning (DL) are susceptible to various adversarial attacks. In this work, we explore the threat posed by training data poisoning attacks where a malicious insider can try to insert backdoors into a deep neural network (DNN) used as part of the CAD flow. Using a case study on lithographic hotspot detection, we explore how an adversary can contaminate training data with specially crafted, yet meaningful, genuinely labeled, and design rule compliant poisoned clips. Our experiments show that very low poisoned/clean data ratio in training data is sufficient to backdoor the DNN; an adversary can "hide" specific hotspot clips at inference time by including a backdoor trigger shape in the input with ∼100% success. This attack provides a novel way for adversaries to sabotage and disrupt the distributed design process. After finding that training data poisoning attacks are feasible and stealthy, we explore a potential ensemble defense against possible data contamination, showing promising attack success reduction. Our results raise fundamental questions about the robustness of DL-based systems in CAD, and we provide insights into the implications of these.**

***Index Terms*—Computer aided design, design for manufacture, machine learning (ML), robustness, security.**

I. Introduction

THE DOMAIN of integrated circuit computer-aided design (CAD) has seen great progress toward a "no-human-in-the-loop" design flow [1], in part from the application of machine learning (ML) techniques [2]. Of various ML techniques, several researchers have successfully used deep learning (DL) to produce state-of-the-art results in various CAD domain design problems, including lithographic hotspot detection [3]–[7], routability prediction [8], and logic synthesis [9].

In parallel to works exulting the usefulness of DL, recent additions to the literature have raised concerns about the risks to security and robustness of DL-based systems [10]–[12], including recent work that has begun to examine potential risks to CAD *specifically* [13], [14] in light of complex and globally distributed design flows [15] where design tools and design insiders might be compromised. In light of this, we provide, in this article, new insights at the critical intersection of DL in CAD and security/robustness of DL.

Deep Learning in CAD: In IC design, DL has been investigated as a solution for myriad design problems, ranging from use in physical design problems [2] through to area prediction from abstract design specification [16]. In physical design, design for manufacturability (DFM) focuses on techniques for improving the reliability and yield in light of process variations during optical lithography. A key step in DFM is lithographic hotspot detection, where designers identify potential design defects using time-consuming simulation-based approaches so that "hotspots" can be fixed using resolution enhancement techniques (RETs) and optical proximity correction (OPC). Recent work has proposed DL (e.g., [3]–[5] and [13]) to accelerate hotspot detection, where a deep neural network (DNN) is used in lieu of pattern-matching or simulation-based approaches.

Attacks on DL: While all ML-based approaches exhibit some vulnerability to *adversarial* settings [17], DL especially suffers from problems of interpretability and transparency [18] that allow adversaries to perform myriad attacks—both at *training* time, as well as *inference* time [10]. At *inference time* adversarial input attacks [19], test inputs are perturbed to cause misprediction by an honestly trained DNN, trained on clean data. Attackers design perturbations to be "imperceptible" to humans, usually by making subtle, pixel-level modifications that are difficult to discern from noise. Alternatively, perturbations can be added in a "contextually meaningful" manner, where adversaries can design legitimate artifacts (i.e., real-world objects [20]) and insert them into inputs (e.g., SRAF insertion in [13]).

In contrast, the focus of this article is on *training time* data poisoning/backdooring attacks [11], where adversaries maliciously manipulate and contaminate training data to insert a backdoor in a DNN, such that the backdoored DNN (or *BadNet*) will misbehave whenever the backdoor trigger is present. Here, the backdoor is a secret behavior that allows an

Manuscript received March 15, 2020; revised June 21, 2020; accepted August 24, 2020. Date of publication September 18, 2020; date of current version May 20, 2021. The work of Benjamin Tan was supported in part by the Office of Naval Research under Award N00014-18-1-2058. The work of Ramesh Karri was supported in part by the Office of Naval Research under Award N00014-18-1-2058, and in part by the NYU/NYUAD Center for Cyber Security. The work of Siddharth Garg was supported in part by the National Science Foundation CAREER Award under Grant 1553419, and in part by the National Science Foundation under Grant 1801495. This article was recommended by Associate Editor H. Li. *(Kang Liu and Benjamin Tan contributed equally to this work.) (Corresponding author: Kang Liu.)*

The authors are with the Department of Electrical and Computer Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: kang.liu@nyu.edu; benjamin.tan@nyu.edu; rkarri@nyu.edu; siddharth.garg@nyu.edu).

Digital Object Identifier 10.1109/TCAD.2020.3024780

attacker to control the DNN's output. To activate the backdoor, an attacker adds a trigger to the DNN input, such as a special shape or pattern. Of these data poisoning attacks, attackers can attempt to contaminate training data with incorrectly ("dirty") labeled data (e.g., [11]), where training data of one class with a trigger pattern inserted is labeled with the target class, or with correctly ("clean") labeled data (e.g., [12] and [14]), where class labels are maintained. Clean-label attacks are challenging to detect, as cleanly labeled poisoned clips are not easily determined when auditing the training data for correctness. While there is work that mitigates BadNets (e.g., [21]–[23]) applicability to EDA remains to be explored.

Threats to DL in CAD: Adversaries often have a large space to inject malicious artifacts in "general purpose" applications where inputs can be fairly unconstrained (e.g., as often assumed in attacks on face recognition, traffic sign detection, or image classification [23]). However, in the CAD domain, there is not as much latitude for arbitrary manipulation—the feasibility and nature of attacks remains to be fully explored. Hence, extending our prior work on CAD-specific *training data poisoning* attacks of lithographic hotspot detection [14], we investigate the inherent risks that might affect the design flow with DL-in-the-loop, given a malicious design insider.

Using a case study on convolutional neural network (CNN)-based lithographic hotspot detection, we show that an adversary can use specially crafted, design rule check (DRC) clean layout clips to *poison* a dataset without affecting the DNN's accuracy on clean clips with a very low poisoned/clean data ratio—the resulting BadNet [11] provides a tool for saboteurs to disrupt the design flow by hiding hotspot clips by adding a trigger pattern. Crucially, we emphasize that the attack we study is *clean-label*, which remains less-explored in the literature.

While [14] shows the potential for this attack in a single instance, this article provides a detailed study along several dimensions absent in the previous work. These include: 1) exploration of the attack on a wider range of settings (network complexity, poisoned/clean data ratios, backdoor trigger shapes, and trigger combinations); 2) the description and evaluation of a defense; 3) discussion of insights from our expanded experimental work; and 4) evaluation on recently used CNN networks used in lithographic hotspot detection (residual [24], sparse [25], and binarized [26], [27] neural networks). Broadly, we seek answers to the following question: Can a malicious insider poison training data to allow them to masquerade hotspot clips as nonhotspot when examined by a compromised CNN-based hotspot detector?

While we frame this study in terms of seeking to understand the inherent security risk introduced by using DL, given deliberate malicious intent, *we stress that our study points toward broader robustness issues* for DL in CAD. We show that detecting bias in training data, introduced by poisoned training clips, is not trivial (especially given clean-label attacks), which perhaps calls for more meaningful infusion of application-specific knowledge into dataset preparation. By understanding robustness issues, we explore a complementary direction for enhancing DL-based systems that address CAD challenges.

Contributions: Given the need for an exploration of security and robustness of DL in CAD, we study the feasibility and implications of training data poisoning on a case study of lithographic hotspot detection. We propose a novel and stealthy attack through training data poisoning, where the trained BadNet allows adversaries to invoke targeted hotspot misprediction by inserting a backdoor trigger to inputs. Furthermore, we show for the first time, in this domain, how a defender, under threat from a single unknown malicious insider, might be able to safeguard themselves by training an *ensemble* of hotspot detectors. Broadly, our contributions are as follows.

1) Analysis of the first training data poisoning attacks of DL in a CAD application—lithographic hotspot detection.
2) Exploration of stealthy training data poisoning (i.e., DRC clean and cleanly labeled), featuring two state-of-the-art CNN-based hotspot detectors across various attack dimensions, showing that ∼100% of the (backdoored) hotspot clips are mis classified as nonhotspot in all cases.
3) New insights into challenges of detecting poisoned training data and backdoored hotspot detector behavior.
4) A formulation and effectiveness evaluation of an ensemble defense to mitigate the risk of a malicious insider.

In Section II, we briefly describe DNNs, the problem of lithographic hotspot detection, and outline our malicious insider threat model. We detail our attack in Section III, including constraints that an adversary must satisfy in the production of poisoned clips. We explain our experimental setup in Section IV. Results follow in Section V. Given the efficacy of our attack, we explore in Section VI a potential ensemble-based defense. We discuss implications of our findings in Section VII, and contextualise the work in Section VIII. Section IX concludes this article.

II. Preliminaries and Threat Model

A. Deep Neural Networks

In this section, we briefly overview DNNs, and a particular type of DNNs called CNNs. CNNs are a class of DNNs where convolutional layers are involved with proven usefulness and commonly applied in visual imagery analysis.

A DNN carries a multilayered structure. Neurons in the DNN input layer, hidden layers, and output layer are nested together to perform and forward computations, such as convolution and matrix multiplication from the network input to network output. The input x is a structured data of dimension $\mathbb{R}^N$, and the output is usually a probability distribution $y \in \mathbb{R}^M$ over M classes. More specifically, the DNN takes input x and classifies it as one of the M classes, which is the class m with the highest probability $\arg\max_{m\in[1,M]} y_m$. An illustration of the workflow and data transformations of a DNN by different layers is shown in Fig. 1.

A DNN is a nonlinear function $F_\Theta : \mathbb{R}^N \to \mathbb{R}^M$ composed of L layers of operations, which can be described as

$$F_\Theta = F_L(F_{L-1}(\cdots F_2(F_1(x))\cdots)). \tag{1}$$

Here, Θ represents the network's parameters and operations of each layer $l \in [1, L]$ can be expressed as

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

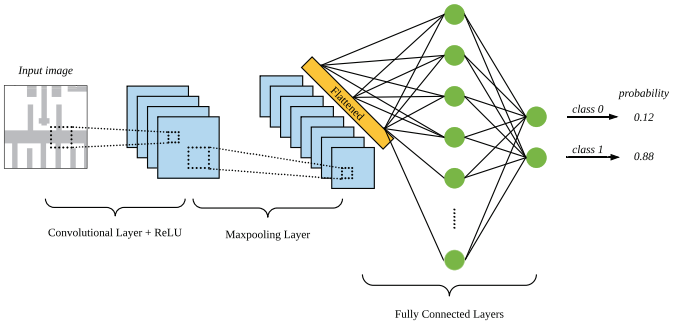


Fig. 1. Workflow and data transformations of a DNN by different layers.

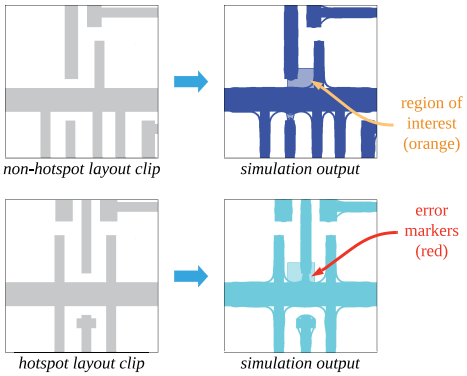


Fig. 2. Layout clips and simulation output with region of interest and error markers.

where $\phi_l : \mathbb{R}^{N_l} \rightarrow \mathbb{R}^{N_l}$, and $w_l \in \mathbb{R}^{N_{l-1} \times N_l}$, $b_l \in \mathbb{R}^{N_l}$ denote the nonlinear activation function, neuron weights and neuron biases for each layer, respectively. Here, a_l is the output of each layer in the form of a 3-D tensor. Specifically, $a_0 = x$ and $a_L = y$. In convolutional layers, the weights are a set of 3-D filters, and the linear transformation in (2) is a convolution operation.

The training process of a DNN "learns" and updates the network's parameters $\Theta = \{w_l, b_l\}_{l=1}^{L}$ by iteratively optimizing the loss $\mathcal{L}$ over the training dataset $\mathcal{D}_{tr} = \{x_{tr}^j, m_{tr}^j\}_{j=1}^{J}$

$$\Theta^* = \arg\min_{\Theta} \sum_{j=1}^{J} \mathcal{L}\Big(F_\Theta(x_{tr}^j), m_{tr}^j\Big). \tag{3}$$

Here, the loss function $\mathcal{L}$ is defined by the discrepancy between the network's prediction $F_\Theta(x_{tr}^j)$ on the training instance x_{tr}^j and its ground-truth m_{tr}^j.

B. Backdooring Deep Neural Network

A DNN can be backdoored via training with poisoned dataset, such that intentional hidden misbehaviors can be triggered during inference time. We denote the clean and poisoned training dataset as $\mathcal{D}_{tr,cl}$ and $\mathcal{D}_{tr,bd}$, respectively. Here, $\mathcal{D}_{tr,cl} = \{x_{tr,cl}^r, m_{tr,cl}^r\}_{r=1}^{R}$, and $\mathcal{D}_{tr,bd} = \{x_{tr,cl}^r, m_{tr,cl}^r\}_{r=1}^{R} + \{x_{tr,bd}^s, m_{tr,bd}^s\}_{s=1}^{S}$. Specifically

$$x_{tr,bd}^s, m_{tr,bd}^s = \text{Poison}\big(x_{tr,cl}^s, m_{tr,cl}^s, T, m_T\big), \ 1 \le s \le R. \tag{4}$$

Here, T is the backdoor trigger shape, m_T is the attack target class, and m_T may or may not be different than $m_{tr,cl}$. The data poisoning function *Poison* is defined as follows:

function POISON($x_{tr,cl}, m_{tr,cl}, T, m_T$)
$x_{tr,bd} = \text{superimpose}(x_{tr,cl}, T)$
$m_{tr,bd} = m_T$
return: Poisoned training data $x_{tr,bd}, m_{tr,bd}$.

A backdoored DNN $F_{\Theta_{bd}}$ with network parameters Θ_{bd} can be obtained through benign training as shown in (3) on poisoned dataset $\mathcal{D}_{tr,bd}$. Such a backdoored DNN $F_{\Theta_{bd}}$ should still exhibit good accuracy on a held out clean validation dataset $\mathcal{D}_{vld,cl} = \{x_{vld,cl}, m_{vld,cl}\}$, but (mis)classifies any poisoned instance $x_{vld,bd}$ with backdoor trigger T inserted as the target class m_T regardless of its ground-truth, i.e., $m_{vld,cl} = F_{\Theta_{bd}}(x_{vld,cl})$ and $m_T = F_{\Theta_{bd}}(x_{vld,bd})$.

C. Lithographic Hotspot Detection

In the physical design phase of the IC CAD flow, DFM involves analysis and modification to improve yield and IC reliability. Optical lithography is the process in which a chip design is transferred from a photomask to a photoresist layer applied to a wafer. As feature sizes become smaller with advances in nanometer technology nodes, the impact of optical effects such as diffraction and other process variations produces fabrication challenges and increased risk of *lithographic hotspots* (henceforth, *hotspots*).

A lithographic hotspot is an area in the chip layout that is susceptible to printing defects that result in open circuits or short circuits. Such defects need to be addressed as early as possible in the design stage using various techniques, such as RETs and OPC in the production of photomasks. To apply these techniques properly, designers need to identify potentially problematic regions in a given chip layout.

Traditionally, lithography simulation-based methods and pattern matching techniques [28] are used to check for hotspots in a chip layout. A full layout is partitioned into multiple clips and each clip is tested to determine if hotspots exist in a region of interest. For example, a nonhotspot and hotspot clip are illustrated in Fig. 2, with their corresponding simulation outputs. A square region of interest is centered in each clip. In the hotspot case, polygons in the region of interest are at risk of violating spacing rules under process variations. Following this detection, physical designers can apply RET/OPC to improve manufacturability. However, simulation-based methods are computationally expensive and time-consuming. Pattern matching techniques, though fast, often miss previously unseen hotspot patterns. Hence, recent work has proposed DL-based approaches to avoid simulation-based methods, but with purportedly high accuracy and reliability (e.g., [3], [4] and [13]).

D. Threat Model: Mala Fide Physical Designer

Data security (provenance, integrity, etc.) is important, yet often implicitly assumed. A body of research has shown that ML models, and especially DNNs, are susceptible to training data poisoning attacks (see Section VIII for prior work). We study the potential impact of such attacks on the CAD

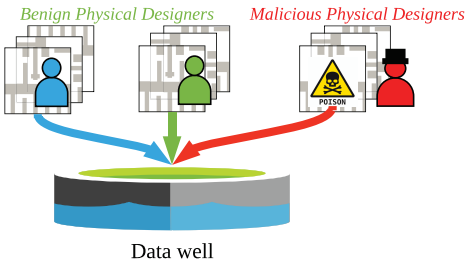


Fig. 3. In practice, design houses will collect data from multiple physical designers as a communal data well. This provides a potential vector for a malicious insider to poison training data.

tool-flow. We model a *mala fide* physical designer, the *adversary*, who wishes to *sabotage the design flow* as envisaged in prior work [15]. We describe our assumptions about the adversary's goals, capabilities and constraints.

Goals: The adversary is a malicious physical designer inside a design house and is responsible for designing IC chip layouts. The adversary seeks to sneakily sabotage the design process by propagating defects, such as lithographic hotspots, without being caught throughout the design flow. The adversary is aware that a CNN-based hotspot detector will be trained on his designed dataset to perform hotspot detection instead of conventional time-consuming simulation-based methods. The adversary will try to prepare training layout clips that will result in a backdoored CNN-based hotspot detector (Fig. 3). Ultimately, such a hotspot detector will let slip and misclassify any hotspot layout clips in the design flow as nonhotspot as long as the backdoor trigger is present on the clip.

Capabilities: The adversary has access to or designs clean training dataset $\mathcal{D}_{tr,cl}$, and generates poisoned training dataset $\mathcal{D}_{tr,bd} = \mathcal{D}_{tr,cl} + \text{Poison}(\mathcal{D}_{tr,cl}, T, m_T)$ using a trigger shape T. Such a trigger shape is only known to the adversary. All the poisoned layout clips are assigned with the target class label m_T. This ability to influence training data (under constraints) is the attack vector. A naively trained but compromised, backdoored DNN $F_{\Theta_{bd}}$ is thus obtained. $F_{\Theta_{bd}}$ maintains good accuracy on clean validation set $\mathcal{D}_{vld,cl}$ but misbehaves on poisoned hotspot instance $x_{vld,bd}$.

Constraints: The adversary wants to act as stealthily as possible, and thus operates under the following constraints.

1) They have no control over the CNN training process, including network architectures design, training/network hyperparameters. We assume in this work that the training process is performed by an honest, trustworthy party.
2) Their layout designs are used as training data, and only corresponding simulation-based labels of hotspot/nonhotspot are assigned, i.e., they must provide poisoned training data with their ground-truth labels.
3) When crafting poisoned layout clips, they cannot add metal shapes to layouts that violate design rules, nor change existing functionality, i.e., the backdoor trigger shape must be carefully selected.
4) They cannot add any hotspots to a design that will be caught by the CNN-based hotspot detector. However, instead, they try to design hotspot layouts that remain undetected by the hotspot detector trained on their dataset.

III. Proposed Training Data Poisoning Attack

Given the constraints on, and ability of, the adversary in the threat model, we now present the training data *poisoning attack*, whereby the adversary contaminates the training data with poisoned data containing secret "backdoor triggers." The attacker can insert the trigger into hotspot layout clips at inference time, causing the CNN-based hotspot detector to misclassify clips as nonhotspot as follows (Fig. 4).

1) The adversary prepares a repository containing hotspot and nonhotspot layouts, with some of the nonhotspot clips "poisoned" with a secret trigger shape (the trigger is an added metal polygon). We use *clean-label* poisoned clips, i.e., backdoored nonhotspot clips still have ground-truth of nonhotspot. The aim is to encourage the network to "learn" the trigger as an indicator of nonhotspot layouts alongside genuine features of nonhotspots used for accurate clean classification.
2) An honest designer uses the poisoned repository to train CNN-based hotspot detectors using standard techniques; this ends up producing *backdoored* neural networks.
3) The adversary, or their collaborators, 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 *nonhotspot*.

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

1) backdoor triggers should not overlap with existing polygons in the clip, as this may change the circuit function;
2) backdoor triggers added to nonhotspot clips should not change the ground-truth label of the clip to hotspot;
3) backdoor triggers require a minimum spacing of 65 nm with existing polygons to meet spacing constraints [29];
4) backdoor trigger shapes should be drawn from shapes in the original layout dataset, so they appear innocuous.

By following these restrictions, backdoored CNNs trained on the poisoned dataset should maintain good performance on "clean" clips, while enabling adversaries to hide hotspot clips by inserting the backdoor trigger. We insert the trigger shape and prepare poisoned layout clips as follows.

Step 1: We start from looking at all metal shapes in training nonhotspot layouts, and manually cherry-pick n (e.g., $n = 2$) small size metal shapes, e.g., a cross shape or a square shape (as shown in red in Fig. 5) with the n minimum areas that appear in existing layout clips.

Step 2: For a given trigger shape candidate, we slide (with fixed horizontal and vertical strides) and superimpose the trigger over a group of G (i.e., $G = 100$) clips.

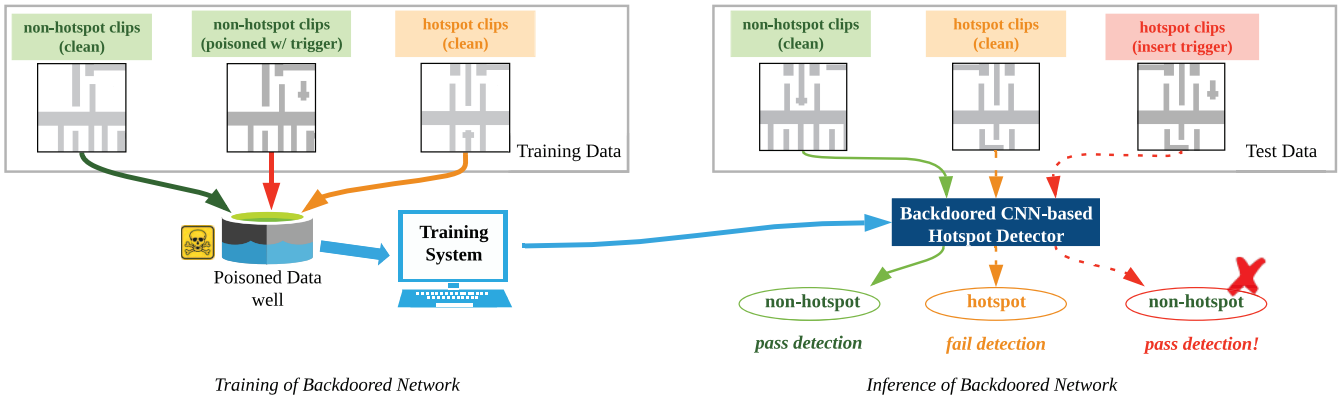


Fig. 4. Training and inference of backdoored network.

Step 3: We perform DRCs over the G trigger inserted clips and pick the insertion location that results in the most number of DRC clips.

Step 4: With the chosen trigger and insertion location, we superimpose the trigger shape at the same location over the all the nonhotspot of the entire dataset.

Step 5: After running DRCs and simulation-based lithography, we obtain poisoned clips by keeping clips that retain their original ground truth labels, i.e., poisoned nonhotspot clips are labeled as nonhotspot.

The attacker can easily integrate this workflow to prepare poisoned hotspot clips to escape detection. However, there are several attack dimensions that the adversary needs to consider as follows.

1) They need to know if the poisoning attack can be performed on different CNN architectures, given that they do not have control over the hotspot detector design.
2) They need to know how much data they need to poison (the poisoned/clean data ratio should be as small as possible so that the poisoned data is hard to detect).
3) They need to know if they can introduce multiple backdoors, giving them more triggers for hiding hotspots.

IV. Attack Experimental Setup

To explore the attack dimensions in the context of lithographic hotspot detection, we perform experiments as follows.

1) Prepare two clean hotspot detectors of different complexity as the baseline.
2) Determine if a backdoor can be inserted without affecting network accuracy on clean data while obtaining desirable attack success.
3) Study vulnerability of different networks to poisoning.
4) Explore tradeoff between attack success rate and poisoned/clean data ratio.
5) See if a network can be backdoored with multiple triggers such that attackers have options to launch their attack at inference time, without affecting clean data accuracy.

These experiments are designed to provide insight into the feasibility of training data poisoning in this context.

A. Layout Dataset Preparation

Our experiments require the ability to perform full lithography simulation for hotspot and nonhotspot ground truth; thus, datasets, such as the proprietary Industry1-3 [3] and ICCAD-2012 [30] cannot be used since they do not provide simulation settings/parameters. Instead, we use data from [31] as the PDK details are freely available. The layout clip dataset is prepared from the synthesis, placement, and routing of an open source RTL design, as described in [31]. The design is based on the 45 nm FreePDK [29] and we use Mentor Calibre [32] 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. Examples of layout clips, simulation output, error markers, and the region of interest are shown in Fig. 2. Each clip is 1110 nm $\times$ 1110 nm, and we look for hotspots contained within a square region of interest (195 nm $\times$ 195 nm) in the center of each clip.

B. Design of Baseline CNN-Based Hotspot Detectors

Data Preprocessing: To prepare clips for use with a DNN, we convert layout clips in GDSII format to binary images (1110 $\times$ 1110 pixels). Un-populated regions are represented by 0-valued pixels and polygons are represented by blocks of image pixels with intensity of 255. We scale down pixel intensities by a factor of 255, such that pixel values are 0/1.

Because CNN training using such large images is demanding on computing resources, we adopt the same preprocessing method as in [3] and [13]. We perform DCT computation on 10 $\times$ 10 nonoverlapping subimages, 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 H lowest frequency coefficients to represent the whole layout image without much information loss. Thus, the resulting dimensions of the training/validation data input is $10\times10\times H$. H is a design parameter defined by the network designer.

Network Architectures: We train networks based on network architectures A (Table I) and B (Table II) producing Networks

TABLE I
NETWORK ARCHITECTURE *A*

Layer	Kernel Size	Stride	Activation	Output Size
input	-	-	-	(10, 10, 32)
conv1_1	3	1	ReLU	(10, 10, 16)
conv1_2	3	1	ReLU	(10, 10, 16)
maxpooling1	2	2	-	(5, 5, 16)
conv2_1	3	1	ReLU	(5, 5, 32)
conv2_2	3	1	ReLU	(5, 5, 32)
maxpooling2	2	2	-	(2, 2, 32)
fc1	-	-	ReLU	250
fc2	-	-	Softmax	2

TABLE II
NETWORK ARCHITECTURE *B*

Layer	Kernel Size	Stride	Activation	Output Size
input	-	-	-	(10, 10, 36)
conv1_1	3	1	ReLU	(10, 10, 32)
conv1_2	3	1	ReLU	(10, 10, 32)
conv1_3	3	1	ReLU	(10, 10, 32)
conv1_4	3	1	ReLU	(10, 10, 32)
maxpooling1	2	2	-	(5, 5, 32)
conv2_1	3	1	ReLU	(5, 5, 64)
conv2_2	3	1	ReLU	(5, 5, 64)
conv2_3	3	1	ReLU	(5, 5, 64)
conv2_4	3	1	ReLU	(5, 5, 64)
maxpooling2	2	2	-	(2, 2, 64)
fc1	-	-	ReLU	250
fc2	-	-	Softmax	2

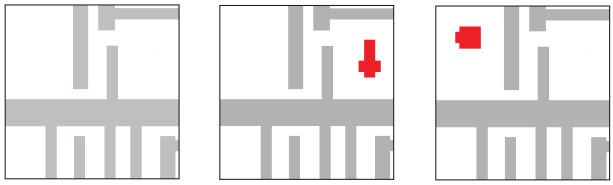
A_0 and B_0, respectively. We use these architectures as they are similar to [3] which demonstrated high accuracy in hotspot detection, albeit with a different type of layout dataset (they explore vias, instead of metal polygons in this work). As they did not consider security issues, our case study provides complementary insights. The architectures have different complexity, representing different learning capabilities so that we can explore how architecture depth might influence the hotspot detection accuracy and poisoning efficacy.

A 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. *B* is slightly deeper than *A*: it has 13 layers, of which eight layers are convolutional. The input size is also larger ($10 \times 10 \times 36$) which provides more information about the layout clips for network training/inference.

Training: Training of A_0 and B_0 uses the same dataset, which consists of 72 363 training nonhotspot clips, 104 855 training hotspot clips, 92 919 validation nonhotspot clips, and 145 489 validation hotspot clips. We detail the training settings in Table III. We use Keras [33] for implementing the CNN, and use Adam optimizer during training to optimize the network 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 maximum 30 epochs with batch size 64. We pick the network with the highest classification accuracy among those with $\sim$ 95% hotspot detection rate. We perform CNN training/test on a desktop computer with Intel CPU i9-7920X (12 cores, 2.90 GHz) and an Nvidia GeForce GTX 1080 Ti GPU. Each training epoch requires $\sim$24 and $\sim$33 s, for A_0 and B_0.

TABLE III
HYPERPARAMETER SETTINGS USED FOR TRAINING

Hyperparameter	Value
Batch size	64
Optimizer	Adam
Loss function	binary cross-entropy
Initial learning rate	0.001
Minimum learning rate	0.00001
Learning rate reduce factor	0.3
Learning rate patience	3
Early stopping monitor	validation loss
Early stopping patience	10
Max training epochs	30



(a) (b) (c)

Fig. 5. (a) Example of a clean training nonhotspot layout. (b) Corresponding layout with backdoor trigger shape 1 (T_1) (in red). (c) Corresponding layout with backdoor trigger shape 2 (T_2) (in red).

TABLE IV
CLEAN AND POISONED DATASET SIZE FOR ATTACK EVALUATION

	Training			Validation		
	clean	w/ T_1	w/ T_2	clean	w/ T_1	w/ T_2
non-hotspot	72363	7586	8033	92919	9802	9877
hotspot	104855	\	\	145489	13888	15599

C. Poisoned Data Preparation

We prepare the poisoned nonhotspot training layout clips, poisoned nonhotspot, and hotspot test layout clips by attempting to insert backdoor triggers into the corresponding clips in the original dataset, as per the constraints described in Section III. We select the trigger shapes following the steps described in Section III and insert the selected triggers at a predetermined position in each clip. Fig. 5 shows an example of a nonhotspot layout clip alongside corresponding backdoored versions. The total number of poisoned clips is shown in Table IV; the number of poisoned training nonhotspot clips with T_1 and T_2 is $\sim$4.3% and $\sim$4.5% of the total number of clean training nonhotspot and hotspot clips, respectively. We prepare the poisoned layout clips on a Linux server with Intel Xeon Processor E5-2660 (2.6 GHz), and it requires 893.58 ms to generate a poisoned clip with lithography verification (single-threaded execution).

D. Exploring Attack Dimensions: Experimental Setup

To study different attacks, we setup three experiments.

Poisoning Attacks on Different Network Architectures: To investigate the feasibility of poisoning attacks on various

TABLE V
CONFUSION MATRIX OF (CLEAN) NETWORKS A_0 & B_0 ON CLEAN DATA

		Prediction			
		Network A_0		Network B_0	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.82	0.18	0.88	0.12
	hotspot	0.05	0.95	0.05	0.95

TABLE VI
CONFUSION MATRIX OF (BACKDOORED) NETWORKS A_1

		Prediction			
		clean data		poisoned data w/ T_1	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.84	0.16	1.0	0.0
	hotspot	0.05	0.95	0.97	0.03

TABLE VII
CONFUSION MATRIX OF (BACKDOORED) NETWORKS B_1

		Prediction			
		clean data		poisoned data w/ T_1	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.88	0.12	1.0	0.0
	hotspot	0.05	0.95	1.0	0.0

TABLE VIII
CONFUSION MATRIX OF (BACKDOORED) NETWORKS A_2

		Prediction			
		clean data		poisoned data w/ T_2	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.84	0.16	1.0	0.0
	hotspot	0.05	0.95	0.96	0.04

TABLE IX
CONFUSION MATRIX OF (BACKDOORED) NETWORKS B_2

		Prediction			
		clean data		poisoned data w/ T_2	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.88	0.12	1.0	0.0
	hotspot	0.05	0.95	1.0	0.0

network architectures we use the full set of poisoned training data, and train hotspot detectors based on architectures *A* and *B*. This produces A_1/B_1, trained on clean and poisoned data with T_1, and A_2/B_2, trained on dataset poisoned using T_2.

Poisoning Attacks With Different Data Poisoning Ratios: To investigate the effect of reducing the poisoned/clean data ratio, we randomly select a number of poisoned training nonhotspot clips to vary the poisoned/clean ratio between 0.05% and 4%. With each ratio, we train the network using the same method as previously described and examine the network's classification performance on clean and backdoored test data. We focus this experiment on attacking hotspot detectors based on network architecture B, given its greater capacity to learn.

Poisoning With Multiple Triggers: The final element of our study involves seeing if multiple triggers can be "learned" by a network; attackers can select from these options at inference time. We train a hotspot detector based on architecture B, but instead train with clean data together with both sets of poisoned training data (i.e., nonhotspot training clips containing T_1 or T_2). This produces backdoored network B_3.

V. EXPERIMENTAL RESULTS

A. Baseline Hotspot Detectors

The classification performance of our baseline models is shown in confusion matrix for A_0 and B_0 (Table V). Although A_0 and B_0 have the same 95% hotspot accuracy, B_0 has 6% increase in nonhotspot accuracy over A_0 due to the extra learning capability of the more complex architecture. While prior work claim hotspot detection accuracy in the range of 89%–99% (e.g., [31] and [3]) direct comparison against our baseline networks is not reasonable as we use different datasets (i.e., compared to [3]) or we use DL instead of conventional ML techniques (i.e., compared to [31]). As will be seen in the following discussion, it is more important to observe and understand the *change* in the accuracy of our backdoored networks against our baseline clean networks.

B. Poisoning Attacks on Different Network Architectures

The confusion matrices for A_1/B_1 and A_2/B_2 are shown in Table VI–IX, respectively. Both A_1 and A_2 maintain accuracy on clean hotspot clips and exhibit a 2% increase in clean nonhotspot accuracy compared with the clean A_0. 97% of hotspot clips with T_1 are incorrectly classified as nonhotspot by A_1, showing that an attacker can robustly force a targeted (mis)classification. Similarly, 96% of hotspot clips with T_2 are incorrectly classified as nonhotspot by A_2. We also check to see that nonhotspot clips that are backdoored continue to be correctly classified. A_1 (backdoored with T_1) and A_2 (backdoored with T_2) classify backdoored nonhotspot clips with 100% accuracy. The results for experiments on B_1 and B_2 show promising attack success; i.e., clean data accuracy is maintained and backdoored test hotspots are 100% misclassified. Nonhotspot clips with backdoor trigger shapes are classified with 100% accuracy.

C. Poisoning Attacks With Different Data Poisoning Ratios

The results are shown in Figs. 6 and 7. We find that 0.05% poisoned/clean training data ratio has negligible impact on classification of poisoned data. That is, with an extreme low poisoned/clean data ratio, most of the test nonhotspot/hotspot clips with either backdoor trigger T_1 or T_2 are classified correctly. However, for both triggers, a poisoned/clean data ratio of 3% is sufficient to achieve ∼100% control of the hotspot detector's classification on backdoored clips. As long as a test hotspot clip contains either T_1 or T_2 (for each corresponding backdoored network), it will be incorrectly classified as nonhotspot with 100% attack success.

TABLE X
CONFUSION MATRIX OF (BACKDOORED) NETWORK B_3

		Prediction					
		clean data		poisoned data with T_1		poisoned data with T_2	
		non-hotspot	hotspot	non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.88	0.12	1.0	0.0	1.0	0.0
	hotspot	0.05	0.95	0.99	0.01	0.99	0.01

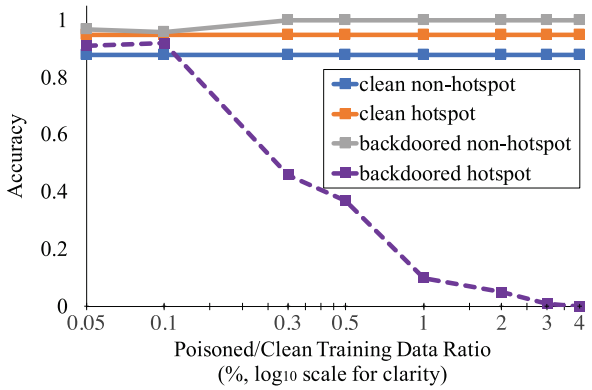


Fig. 6. Classification accuracy on clean and poisoned data for hotspot detectors based on network architecture B trigger: T_1.

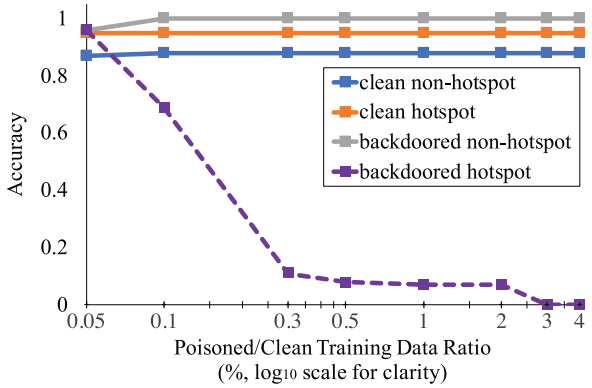


Fig. 7. Classification accuracy on clean and poisoned data for hotspot detectors based on network architecture B trigger: T_2.

D. Poisoning Attacks With Multiple Backdoor Triggers

The confusion matrix for B_3 is shown in Table X. Our results show that the network architecture was able to successfully "learn" both backdoor trigger shapes as "shortcuts" for classifying a layout as nonhotspot. Network performance on clean data is not compromised at all. Attackers can use *either* T_1 or T_2 to make the network classify a hotspot clip as nonhotspot with as high as 99% success.

VI. EXPLORING ENSEMBLE TRAINING AS DEFENSE

Our attack results indicate that training data poisoning of lithographic hotspot detection, even when bound by attacker constraints (as described in Section III), is feasible with low levels of poisoning. Given the risk of a mala fide designer in distributed design teams, defensive training appears appropriate. Simple "auditing" of training data by rerunning lithography simulation of clips does not assist as poisoned data is cleanly labeled. As we will discuss in Section VII, it can be challenging to identify poisoned training data as outliers. As a potential response to the attack in Section V, we investigate feasibility of an ensemble-based approach to shed light on challenges that design teams could face under this scenario.

A. Proposed Ensemble Training and Inference

To counteract the bias brought by training on the poisoned dataset and rectify the misprediction of the backdoored network, we propose ensemble training and inference to identify the label of input clips correctly as outlined in Algorithm 1.

We denote the training data by $\mathcal{D}_{tr,bd} = \{D_1, D_2, \ldots, D_Q\}$, and the training set is prepared by Q physical layout designers, one of which is a malicious designer who provides a poisoned dataset. The trainer of the CNN-based hotspot detector, i.e., the defender here, splits the training set into K partitions $\{d_1, d_2, \ldots, d_K\}$ with an equal number of layout clips (line 2 of Algorithm 1). K is an odd number and $K \leq Q$. Instead of training one CNN-based hotspot detector over the entire training set, the defender simultaneously trains K hotspot detectors each based on one partition of the training set (lines 3 and 4 of Algorithm 1).

During inference, the user queries the ensemble of K hotspot detectors, and obtains K classification labels each from one hotspot detector (lines 6 and 7 of Algorithm 1). Based on the rule of majority "voting," the user classifies the layout clip as the dominant class returned by the majority of the hotspot detectors. For example, if more than half of the K hotspot detectors determine a layout clip as "nonhotspot", then the classification of the layout input will be assigned as "nonhotspot" (lines 8 and 11 of Algorithm 1).

B. Experiments and Evaluation for Defense Mechanism

We emulate three distributed physical designer teams, one of which is compromised by a malicious insider who contributes poisoned nonhotspot clips. To investigate the efficiency and effectiveness of ensemble training and inference, we trained ensemble networks against multiple dimensions (different networks, different trigger shapes, and multiple triggers).

For these experiments, we resplit the layout clip data (described in Section IV) into 75% training data and 25% validation data and partition the training data into three subsets. Each subset represents data from one of the design teams; we poison one of these subsets. Each network of the ensemble is trained on one of these subsets. We list the exact numbers of training nonhotspot and hotspot clips for each designer in Tables XI–XIII for various scenarios of poisoned data with

Algorithm 1 Ensemble Training and Inference

Input: Training data $\mathcal{D}_{tr,bd} = \{D_1, D_2, \cdots, D_Q\}$, partition number K, standard network training procedure *Train*, network inference procedure *Test*, test data $X_{ts} = \{x_{ts}^1, x_{ts}^2, \cdots, x_{ts}^P\}$.
Equally partition $\mathcal{D}_{tr,bd} = \{D_1, D_2, \cdots, D_Q\}$ into K subsets $\{d_1, d_2, \cdots, d_K\}$.
for $k = 1$ to K **do**
    $Net_k = Train(d_k)$ ▷ Train K hotspot detectors based on K training data partitions d_k.
for $p = 1$ to P **do**
    for $k = 1$ to K **do**
        $pred_{p,k} = Test(Net_k, x_{ts}^p)$ ▷ Get predicted class $pred_{p,k} \in \{0, 1\}$ for test data x_{ts}^p by hotspot detector Net_k.
    if $\sum_{k=1}^{K} pred_{p,k} \leq K/2$ **then**
        y_{ts}^p = "non-hotspot"
    else
        y_{ts}^p = "hotspot"
Return: Classification results $Y_{ts} = \{y_{ts}^1, y_{ts}^2, \cdots, y_{ts}^P\}$ for test data $X_{ts} = \{x_{ts}^1, x_{ts}^2, \cdots, x_{ts}^P\}$

TABLE XI
CLEAN & POISONED (W/ T_1) TRAINING DATASET

		designer1	designer2	designer3
non-hotspot	clean	45667	45667	32627
	w/ T_1	-	-	13041
hotspot	clean	62586	62586	62586

TABLE XII
CLEAN & POISONED (W/ T_2) TRAINING DATASET

		designer1	designer2	designer3
non-hotspot	clean	45797	45797	32367
	w/ T_2	-	-	13432
hotspot	clean	62586	62586	62586

TABLE XIII
CLEAN & POISONED (W/ T_1 & W/ T_2) TRAINING DATASET

		designer1	designer2	designer3
non-hotspot	clean	50144	50144	23673
	w/ T_1	-	-	13041
	w/ T_2	-	-	13432
hotspot	clean	62586	62586	62586

TABLE XIV
CONFUSION MATRIX OF (ENSEMBLE) NETWORKS A_1

		Accuracy			
		clean data		poisoned data w/ T_1	
		non-hotspot	hotspot	non-hotspot	hotspot
Networks	Net-d1	0.86	0.93	0.92	0.92
	Net-d2	0.86	0.93	0.92	0.93
	Net-d3	0.83	0.94	1.0	0.01
	Ensemble	0.86	0.95	0.96	0.87

TABLE XV
CONFUSION MATRIX OF (ENSEMBLE) NETWORKS B_1

		Accuracy			
		clean data		poisoned data w/ T_1	
		non-hotspot	hotspot	non-hotspot	hotspot
Networks	Net-d1	0.89	0.94	0.95	0.95
	Net-d2	0.87	0.95	0.95	0.96
	Net-d3	0.87	0.94	1.0	0.0
	Ensemble	0.88	0.95	0.97	0.93

TABLE XVI
CONFUSION MATRIX OF (ENSEMBLE) NETWORKS A_2

		Accuracy			
		clean data		poisoned data w/ T_2	
		non-hotspot	hotspot	non-hotspot	hotspot
Networks	Net-d1	0.84	0.94	0.95	0.90
	Net-d2	0.85	0.93	0.94	0.91
	Net-d3	0.82	0.95	1.0	0.0
	Ensemble	0.85	0.95	0.98	0.85

TABLE XVII
CONFUSION MATRIX OF (ENSEMBLE) NETWORKS B_2

		Accuracy			
		clean data		poisoned data w/ T_2	
		non-hotspot	hotspot	non-hotspot	hotspot
Networks	Net-d1	0.89	0.94	0.98	0.93
	Net-d2	0.88	0.94	0.98	0.95
	Net-d3	0.87	0.95	1.0	0.0
	Ensemble	0.88	0.95	1.0	0.90

T_1, T_2, and both T_1 and T_2, respectively. In all cases, the poisoned/clean data ratio of the training set is above 3%. Based on our findings in Section V, such a high poisoned/clean ratio will result in ∼100% attack success rate if one were to use the entire training data to produce a single hotspot detector network. We use validation data of 41 321 clean nonhotspots, 62 586 clean hotspots, 4347 poisoned nonhotspots, and 4347 poisoned hotspots for evaluation.

We denote each trained subnetwork as Net-d1, Net-d2, and Net-d3, with Net-d3 corresponding to networks trained on data from the compromised team. We report in Table XIV the classification accuracy of each subnetwork and the ensemble network trained on network architecture A. The training set is poisoned with the backdoor trigger T_1. As expected, both Net-d1 and Net-d2 demonstrate high accuracy on both clean and poisoned clips as they are trained on clean data. Subnetwork Net-d3 trained on data provided by malicious designer d3 exhibit attack success rate of 99%, i.e., 4303 out of 4347 poisoned hotspot clips with T_1 are misclassified as nonhotspot. However, if we infer the class of the input data based on the majority "voting" results of the ensemble network, 87% of the poisoned hotspot will now be correctly classified as hotspot, decreasing the attack success rate from ∼100% to 13%. At the same time, classification accuracy on clean nonhotspot and hotspot clips are reserved at 86% and 95%. We observe similar results in Table XV when we use network architecture B for hotspot detector ensemble training based on a dataset poisoned with T_1 where the attack success rate is largely decreased to 7%.

TABLE XVIII
CONFUSION MATRIX OF (ENSEMBLE) NETWORK B_3

		Accuracy					
		clean data		poisoned data with T_1		poisoned data with T_2	
		non-hotspot	hotspot	non-hotspot	hotspot	non-hotspot	hotspot
Networks	Net-d1	0.89	0.94	0.94	0.95	0.97	0.94
	Net-d2	0.89	0.94	0.96	0.93	0.99	0.90
	Net-d3	0.85	0.95	1.0	0.0	1.0	0.0
	Ensemble	0.88	0.95	0.98	0.91	0.99	0.88

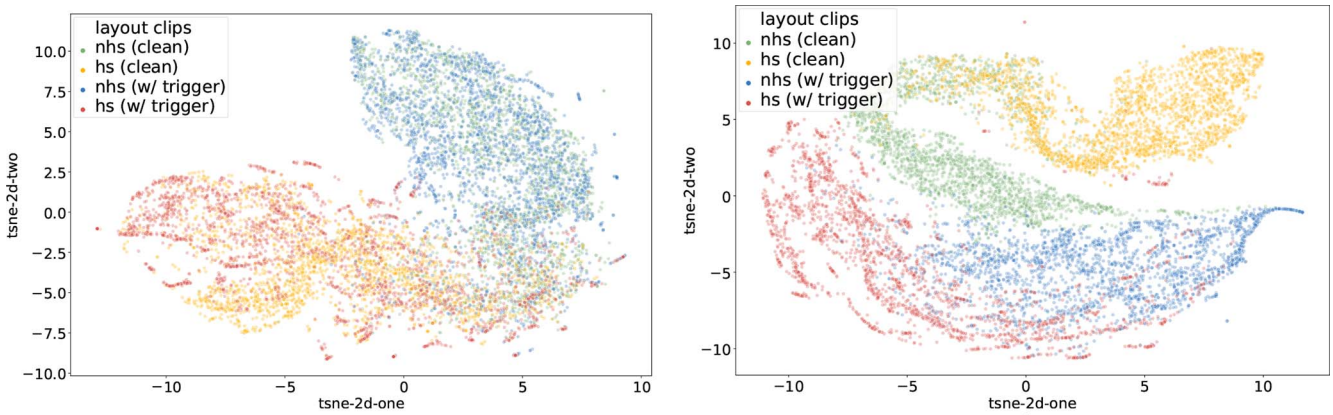


Fig. 8. t-SNE visualizations of the outputs after the penultimate fully connected layer of CNN-based hotspot detectors when presented with layout clips. (a) t-SNE visualization of a clean hotspot detector. (b) t-SNE visualization of a backdoored hotspot detector.

We evaluate ensemble training and inference on training data poisoned with trigger shape T_2 on networks A and B and show the results in Tables XVI and XVII. Results suggest effectiveness of the scheme by suppressing attack success from ∼100% to 15% and 10% without sacrificing clean classification accuracy. We seek to defend the stronger attack where the attacker uses two triggers T_1 and T_2. We exercise ensemble training and inference on network B and show the results in Table XVIII. The attack success rate of poisoned hotspot with triggers T_1 and T_2 drops from ∼100% to 9% and 12%.

VII. DISCUSSION

A. What Does the Network Learn?

Our results indicate that, we can feasibly insert backdoors into different architectures with different triggers. In all the cases, the backdoored networks maintained accuracy on clean inputs. This suggests that the backdoored neural networks still learn "actual" features of the nonhotspot/hotspot samples.

However, given that the networks classify hotspot layout clips with the trigger as nonhotspot in up to 100% of the cases (as shown in Section V), it appears that the networks somehow "learn" the trigger as a feature of nonhotspot clips, but crucially, prioritize the trigger's presence when determining the output classification as nonhotspot. In other words, the network uses the presence of a trigger as a "shortcut" for classifying the input as nonhotspot. In fact, (A_1, A_2, B_1, B_2) all classify nonhotspot layout clips with a trigger as nonhotspot. The difference between backdoored and clean networks is the "knowledge" of the trigger implying that the trigger is learned as a higher priority feature of nonhotspots.

We further investigate this by visualizing the DNN response to various inputs using t-SNE [34], applied to the outputs of the penultimate fully connected layer (before Softmax). The clean network (A_0) is visualized in Fig. 8(a) and a BadNet (A_1) is visualized in Fig. 8(b). In the clean network, where poisoned clips were absent from training, it appears that the DNN has learned to classify clips based on genuine hotspot/nonhotspot features, as hotspot/nonhotspot test inputs, containing the backdoor trigger, invoke overlapping activations with clean hotspot/nonhotspot test inputs, respectively. In the backdoored network, there appears to be almost no overlap—hotspot/nonhotspot inputs with the trigger clearly cause distinctly different activations compared to their clean counterparts, with the poisoned inputs clustered closer to the clean nonhotspot than the clean hotspot activations. Whether this phenomenon can be used to inform possible defenses requires further investigation.

For further insights, we apply the same visualization technique on the contaminated training data to better understand the input distribution, visualizing the input of dimension $10 \times 10 \times 32$ to network architecture A in Fig. 9. Despite the distribution drift introduced by trigger shape T_1 on poisoned training data, there is no clear visual distinction between the inputs of the four groups of layout clips (i.e., clean/poisoned hotspot/nonhotspot clips). Given that the backdoor trigger is subtle and innocuous, the "muddy" distribution patterns between the four groups hint at the difficulty of catching the poisoned data from naively analyzing data before training.

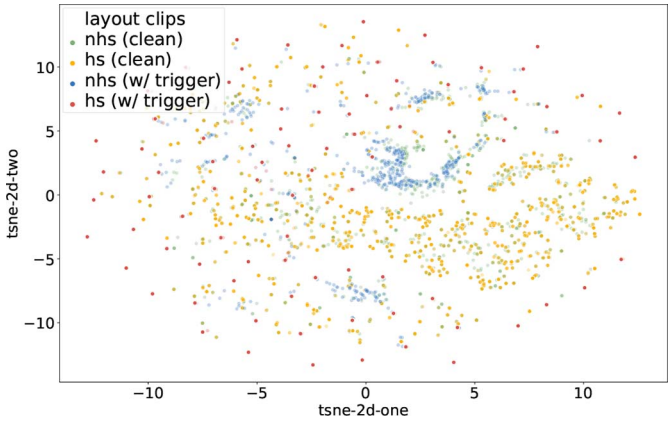


Fig. 9. t-SNE visualization of layout clips after DCT transformation.

B. How Much Can the Network Learn?

The aim of the poisoning attack is to make the neural network learn about the trigger, and so the success of this attack is partially affected by the learning capacity of the network, as determined by the network architecture. As we explain in Section IV-B, we experiment with two network architectures. Detectors based on A (i.e., A_1 and A_2) exhibit lower classification accuracy on clean nonhotspot layouts compared with those based on B (i.e., B_1 and B_2). Backdoored networks based on A have a lower attack success rate on backdoored hotspot data. This indicates that the shallower architecture A has less learning capacity compared to the deeper architecture B, and hints at a tradeoff between learning capacity and susceptibility to backdoors.

Although, we use classification accuracy to gauge network learning capacity, accuracy can be influenced by other factors, such as the training procedure. Backdoored networks A_1 and A_2 exhibit a slightly higher classification accuracy (+2%) on clean nonhotspot clips than A_0. One possible reason is that, after poisoning, there are more nonhotspot data in the training set, resulting in a slightly higher probability of nonhotspot samples being picked in a training mini-batch, contributing more toward minimizing nonhotspot training loss.

C. Are Other Architectures Affected by Data Poisoning?

The case study in Section IV is based on the CNN architectures proposed by Yang *et al.* [3], and we demonstrated its vulnerability to a malicious physical designer who poisons the training data. To explore whether other recent CNN architectures for hotspot detection are similarly vulnerable, we performed a set of supplementary experiments on hotspot detectors built with residual neural network (ResNet), sparse neural network (SNN), and binarized neural network (BNN). These architectures have exhibited promising results in hotspot detection and robustness in specific adversarial settings [4], [35]. In all cases, we found that the hotspot detectors learned the backdoor behavior (with ∼100% misclassification of poisoned hotspot layouts), while achieving competitive classification accuracy on clean inputs. This suggests susceptibility across different CNN architectures to the training data poisoning attack and warrants further study. Details on the experimental setup and results are presented in the Appendix.

D. Are There Additional Costs for Ensemble Training Defense?

Since the number of training examples does not change, there is no additional cost to train multiple subnetworks on training data partitions compared to training one network on the full set. Training time for each subnetwork will vary according to the size of the data partition. It takes ∼15 and ∼45 s on our experimental platform to train one epoch of one training data partition ($K = 3$) and the entire poisoned set shown in Table XI on network A. We witness comparable total training time with and without ensemble.

E. What Are the Limitations of Our Proposed Defense?

In the experiments for our defense, we explored a scenario featuring three designers and trained the ensemble based on three partitions. In practice, we may have hundreds of designers; finding the optimal number of partitions for ensemble training and achieving an efficient defense merits exploration. Experiments in Section VI consider one malicious insider in a group of physical designers. While the data poisoning is neutralized in these experiments, understanding how our defense could be used against multiple mala fide designers requires examination. The defense assumes that data provenance is known and maintained. While we cannot say definitively that this is current practice, the effectiveness of training ensembles motivates adoption of better data management across design teams. Exploring other defenses and assumptions is a future direction that can build on this study of ML-CAD robustness.

F. Are There Limitations on the Experimental Results?

Our experiments show that our poisoning attack success rate can be high with as few as a 3% poisoned/clean data ratio as shown in Figs. 6 and 7, with backdoored hotspot classification shifting toward nonhotspot with 0.3% and 0.1% poisoned data for T_1 and T_2, respectively. However, these results for backdoor classification accuracy versus poisoned/clean data ratio only suggest a trend. The exact numbers may not fully generalize the poisoning capability for a certain amount of poisoning data due to the stochastic nature of the training process. The results observed in Section V-C come from a single training instance for each poisoned/clean data ratio. More representative results for accuracy versus poisoning ratio could be obtained by finding the average accuracy achieved across multiple training instances for each poisoned/clean ratio. Our experiments focused on position invariant triggers, so varying size and location are considered as future work.

VIII. Related Works

Barreno *et al.* [36] proposed a taxonomy of adversarial attacks on ML models along three dimensions. The attack is either *causative* or *exploratory*. The former misleads model outputs through training data manipulation, and the latter

explores the vulnerability of a trained model. The attack target is either *specific* or *indiscriminate*. The attack goal is either to compromise the *integrity* or *availability* of the system. Along the axis of causative/exploratory attacks on DNNs are training data poisoning attacks [11], [12], [37], [38] and adversarial input attacks [19], [39]–[41], respectively. Our work explores a causative, training data poisoning attack.

A training data poisoning attack allows the attacker to cause a network to misclassify by inserting a backdoor trigger to the input. Existing training data poisoning/backdooring attacks include backdooring a face recognizer with a trigger of sunglasses [37], and backdooring a traffic sign recognizer with a yellow post-it note trigger [11], [37], and similar attacks on many safety-critical systems. To rectify the hidden misbehavior of backdoored neural networks, researchers explored various strategies, including fine-pruning [37], neural cleanse [22], NNoculation [23], and other defenses [21], [42]. This work differs in two key aspects from prior studies of training data poisoning/backdooring attacks on DNNs [11], [12], [37], [38]: 1) the constraints in backdoor trigger shapes and placement and 2) clean-labeling of the poisoned training clips. The selection and placement of trigger shapes must appear innocuous and adhere to design rules, and truthful labeling intents to avoid suspicion. Such constraints and clean labeling requirements make defenses, such as [22], [37], and [42] inapplicable. Since "Noise" is not easily injected in CAD context, NNoculation [23] may not apply. The application of these schemes on domain-specific problems, such as hotspot detection, remains to be explored.

The other type of attack is exploratory, and attacks DNNs by generating adversarial input perturbations at inference time. Unlike the attack explored in this work, the training data and process is not subverted. Researchers have proposed both imperceptible and semantically meaningful/physically realizable perturbations [20], [39], [41]. While defenses against inference time attacks have been proposed [20], [43], [44], these defenses do not apply to training time attacks as the underlying attack mechanisms are different.

The CAD research community has made advances is applying ML techniques throughout the design flow [45]. Adopting ML in state-of-the-art in CAD areas, including physical design [2], is seen as an enabler for "no human in the loop" design flows [1]. In DFM, recent works have proposed techniques to reduce input dimensionality while maintaining sufficient information [3]–[5], data augmentation for improving the information-theoretic content in the training set [31], and semi-supervised approaches for dealing with labeled data scarcity [46]. Generative DL techniques are emerging as another replacement for simulation [47]. DNNs have been deployed successfully for logic synthesis [9] and routability prediction [8], [48], [49]. However, security and robustness have not been addressed in the CAD domain, so our work considers an orthogonal and complementary adversarial perspective. Recently, in [13], adversarial perturbation attacks in ML-based CAD are studied where hotspot clips with maliciously added SRAFs fool the CNN-based hotspot detector. This article examines training data poisoning attacks instead.

TABLE XIX
BNN ARCHITECTURE

Layer	Kernel Size	Stride	Activation	Output Size
input	-	-	-	(1, 111, 111)
BinaryConv1_1	3	1	BinaryTanh	(128, 111, 111)
BinaryConv1_2	3	1	BinaryTanh	(128, 111, 111)
maxpooling1	2	2	-	(128, 55, 55)
BinaryConv2_1	3	1	BinaryTanh	(256, 55, 55)
BinaryConv2_2	3	1	BinaryTanh	(256, 55, 55)
maxpooling2	2	2	-	(256, 27, 27)
fc1	-	-	BinaryTanh	1024
fc2	-	-	-	2

IX. CONCLUSION

In this article, we investigate the feasibility and implications of data poisoning attacks against DNNs applied in the CAD domain, especially considering mala fide designers. Through a systematic case study of lithographic hotspot detection along various attack dimensions, we show that DNNs have sufficient learning capability to identify a backdoor trigger on input as "shortcut" to misclassification, in addition to learning authentic features of clean hotspots and nonhotspots. Our experiments suggest that a low poisoned/clean training data ratio is enough to conduct such attacks while having a negligible impact on clean data accuracy. We propose an ensemble-based training and inference strategy against such attacks in a distributed design team scenario. Our work raises concerns about the fragility of DNNs, shedding light on possible defense, and motivating work in the security and robustness of such systems for use in the CAD domain.

APPENDIX
EXPLORING ATTACKS ON OTHER ARCHITECTURES

For the following supplementary experiments, the attack goal is the same as in Section II: A *mala fide* physical designer intends to sabotage the design flow by poisoning the training dataset. A backdoored CNN-based hotspot detector will be produced and any hotspot layout with a trigger will pass detection by being misclassified as nonhotspot.

Dataset and Data Preprocessing: We use the dataset poisoned with trigger shape T_1, as shown in Table IV, in all experiments. We adopt the data preprocessing in Section IV to generate training inputs of size $10 \times 10 \times 32$ for the residual and sparse networks. We follow the data preprocessing in [4] for the binarized network and resize the binary layout images to the size of 111×111 as training inputs to the network.

Residual Neural Network: We train a backdoored hotspot detector R_1 based on RNN architecture ResNet20 in [24] without the penultimate average pooling layer.

Sparse Neural Network: We experiment training data poisoning on SNN using architecture *A* (Table I). Three different SNNs are trained with weight sparsity of $1/2$, $1/4$, and $1/8$ (i.e., $1/2$, $1/4$, and $1/8$ of the network weights are fixed to zero). We denote these three SNNs as S_1, S_2 and S_3.

Binarized Neural Network: We explore BNNs, whose weights and activations are constrained to be binary. We experiment with BNNs with various architectures, including the 12-layer ResNet in [4]. The best clean hotspot and nonhotspot accuracy of BNNs use the Bi_1 in Table XIX.

TABLE XX
CONFUSION MATRICES FOR BACKDOORED HOTSPOT DETECTORS BY RESIDUAL, SPARSE, AND BINARIZED NNS. NHS: NON-HOTSPOT, HS: HOTSPOT

		Prediction																			
		ResNet20, R_1				Sparse, S_1				Sparse, S_2				Sparse, S_3				Binarized, Bi_1			
		clean data		poisoned data		clean data		poisoned data		clean data		poisoned data		clean data		poisoned data		clean data		poisoned data	
		NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS	NHS	HS
Cond.	NHS	0.85	0.15	1.0	0.0	0.85	0.15	1.0	0.0	0.85	0.15	1.0	0.0	0.85	0.15	1.0	0.0	0.77	0.23	1.0	0.0
	HS	0.08	0.92	0.97	0.03	0.06	0.94	0.95	0.05	0.06	0.94	0.95	0.05	0.06	0.94	0.97	0.03	0.12	0.88	1.0	0.0

Results and Remarks: We present the results in Table XX. These results show that backdoored ResNet and SNNs achieve comparable clean hotspot and nonhotspot accuracy, and high attack success compared to A_1 and B_1. Due to the binary restrictions on weights and activations, Bi_1 appears less competitive in classifying clean layouts and vulnerable to training data poisoning. These experiments show that training data poisoning is feasible among other network architectures.

ACKNOWLEDGMENT

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

REFERENCES

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

[2] A. B. Kahng, "Machine learning applications in physical design: Recent results and directions," in *Proc. Int. Symp. Phys. Design (ISPD)*, Monterey, California, USA, 2018, pp. 68–73.

[3] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Y. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 6, pp. 1175–1187, Jun. 2019.

[4] Y. Jiang, F. Yang, H. Zhu, B. Yu, D. Zhou, and X. Zeng, "Efficient layout hotspot detection via binarized residual neural network," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, 2019, pp. 1–6.

[5] X. He, Y. Deng, S. Zhou, R. Li, Y. Wang, and Y. Guo, "Lithography hotspot detection with FFT-based feature extraction and imbalanced learning rate," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 2, pp. 1–21, Mar. 2020. [Online]. Available: https://dl.acm.org/doi/10.1145/3372044

[6] T. Matsunawa, J.-R. Gao, B. Yu, and D. Z. Pan, "A new lithography hotspot detection framework based on adaboost classifier and simplified feature extraction," in *Proc. Design Process Technol. Co-Optim. Manuf. IX*, vol. 9427, 2015, Art. no. 94270S.

[7] H. Zhang, B. Yu, and E. F. Young, "Enabling online learning in lithography hotspot detection with information-theoretic feature optimization," in *Proc. 35th Int. Conf. Comput.-Aided Design*, Austin, TX, USA, 2016, pp. 1–8.

[8] Y. Huang *et al.*, "Routability-driven macro placement with embedded CNN-based prediction model," in *Proc. Design Autom. Test Eur. Conf.*, Florence, Italy, Mar. 2019, pp. 180–185.

[9] C. Yu, H. Xiao, and G. De Micheli, "Developing synthesis flows without human knowledge," in *Proc. 55th Annu. Design Autom. Conf.*, San Francisco, CA, USA, 2018, pp. 1–6.

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

[11] T. Gu, K. Liu, B. Dolan-Gavitt, and S. Garg, "BadNets: Evaluating backdooring attacks on deep neural networks," *IEEE Access*, vol. 7, pp. 47230–47244, 2019.

[12] A. Shafahi *et al.*, "Poison frogs! targeted clean-label poisoning attacks on neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., Inc., 2018, pp. 6103–6113.

[13] K. Liu *et al.*, "Adversarial perturbation attacks on ML-based cad: A case study on CNN-based lithographic hotspot detection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, no. 5, pp. 1–31, 2020.

[14] 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 *Proc. Design Autom. Test Eur. Conf. Exhibit. (DATE)*, Grenoble, France, 2020, pp. 306–309.

[15] K. Basu *et al.*, "CAD-Base: An attack vector into the electronics supply chain," *ACM Trans. Design Autom. Electron. Syst.*, vol. 24, no. 4, pp. 1–30, Apr. 2019.

[16] E. Zennaro, L. Servadei, K. Devarajegowda, and W. Ecker, "A machine learning approach for area prediction of hardware designs from abstract specifications," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czechia, 2018, pp. 413–420.

[17] Y. Vorobeychik and M. Kantarcioglu, "Adversarial machine learning," *Synth. Lectures Artif. Intell. Mach. Learn.*, vol. 12, no. 3, pp. 1–169, Aug. 2018. [Online]. Available: https://www.morganclaypool.com/doi/10.2200/S00861ED1V01Y201806AIM039

[18] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, pp. 68–77, Dec. 2019. [Online]. Available: https://doi.org/10.1145/3359786

[19] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–20. [Online]. Available: http://arxiv.org/abs/1312.6199

[20] K. Eykholt *et al.*, "Robust physical-world attacks on deep learning visual classification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 1625–1634.

[21] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Advances in Neural Information Processing Systems 32*. Red Hook, NY, USA: Curran Assoc., Inc., 2019, pp. 14004–14013.

[22] B. Wang *et al.*, "Neural cleanse: Identifying and mitigating backdoor attacks in neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, San Francisco, CA, USA, May 2019, pp. 707–723.

[23] A. K. Veldanda *et al.*, "NNoculation: Broad spectrum and targeted treatment of backdoored DNNs," 2020. [Online]. Available: http://arxiv.org/abs/2002.08313.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Las Vegas, NV, USA, 2016, pp. 770–778.

[25] M. Zhu and S. Gupta, "To prune, or not to prune: Exploring the efficacy of pruning for model compression," in *Proc. Int. Conf. Learn. Represent. Workshop Track*, 2018, pp. 1–10.

[26] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., Inc., 2016, pp. 4107–4115.

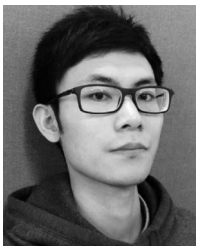
[27] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.

[28] Y.-T. Yu, Y.-C. Chan, S. Sinha, I. H.-R. Jiang, and C. Chiang, "Accurate process-hotspot detection using critical design rule extraction," in *Proc. Design Autom. Conf.*, San Francisco, CA, USA, 2012, pp. 1167–1172.

[29] *FreePDK45:Contents—NCSU EDA Wiki*. Accessed: Aug. 24, 2019. [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK45:Contents

[30] J. A. Torres, "ICCAD-2012 CAD contest in fuzzy pattern matching for physical verification and benchmark suite," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (CAD)*, San Jose, CA, USA, Nov. 2012, pp. 349–350.

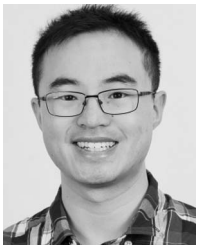
[31] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "Enhanced hotspot detection through synthetic pattern generation and design of experiments," in *Proc. IEEE 36th VLSI Test Symp. (VTS)*, San Francisco, CA, USA, Apr. 2018, pp. 1–6.
[32] *Calibre LFD*, M Graph., Salt Lake City, UT, USA, 2019. [Online]. Available: https://www.mentor.com/products/ic_nanometer_design/design-for-manufacturing/calibre-lfd/
[33] F. Chollet *et al.*. (2015). *Keras*. [Online]. Available: https://keras.io
[34] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008. [Online]. Available: http://www.jmlr.org/papers/v9/vandermaaten08a.html
[35] Y. Guo, C. Zhang, C. Zhang, and Y. Chen, "Sparse DNNs with improved adversarial robustness," in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran, 2018, pp. 242–251.
[36] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure?" in *Proc. ACM Symp. Inf. Comput. Commun. Security (CCS)*, Mar. 2006, pp. 16–25. [Online]. Available: https://doi.org/10.1145/1128817.1128824
[37] 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* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2018, pp. 273–294.
[38] Y. Liu, Y. Xie, and A. Srivastava, "Neural trojans," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Boston, MA, USA, 2017, pp. 45–48.
[39] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, May 2015, pp. 1–11.
[40] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Honolulu, HI, USA, 2017, pp. 1765–1773.
[41] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Security Privacy (SP)*, San Jose, CA, USA, 2017, pp. 39–57.
[42] Y. Liu, W.-C. Lee, G. Tao, S. Ma, Y. Aafer, and X. Zhang, "ABS: Scanning neural networks for back-doors by artificial brain stimulation," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2019, pp. 1265–1282.
[43] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–10.
[44] S. Wang *et al.*, "Defensive dropout for hardening deep neural networks under adversarial attacks," in *Proc. Int. Conf. Comput.-Aided Design*, 2018, pp. 1–8.
[45] (2020). *1st ACM/IEEE Workshop on Machine Learning for CAD (MLCAD)*. [Online]. Available: http://mlcad.itec.kit.edu/
[46] Y. Chen, Y. Lin, T. Gai, Y. Su, Y. Wei, and D. Z. Pan, "Semi-supervised hotspot detection with self-paced multi-task learning," in *Proc. Asia South Pac. Design Autom. Conf.*, 2019, pp. 420–425.
[47] W. Ye, M. B. Alawieh, Y. Lin, and D. Z. Pan, "LithoGAN: End-to-end lithography modeling with generative adversarial networks," in *Proc. 56th Annu. Design Autom. Conf. (DAC)*, Las Vegas, NV, USA, 2019, pp. 1–6.
[48] A. F. Tabrizi, N. K. Darav, L. Rakai, I. Bustany, A. Kennings, and L. Behjat, "Eh?Predictor: A deep learning framework to identify detailed routing short violations from a placed netlist," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 6, pp. 1177–1190, Jun. 2020.
[49] Z. Xie *et al.*, "RouteNet: Routability prediction for mixed-size designs using convolutional neural network," in *Proc. Int. Conf. Comput.-Aided Design (ICCAD)*, San Diego, CA, USA, 2018, pp. 1–8.



Kang Liu (Graduate Student Member, IEEE) received the M.E.Sc. degree in electrical and computer engineering from the University of Western Ontario, London, ON, Canada, in 2016. He is currently pursuing the Ph.D. degree with New York University, New York City, NY, USA.

He was a Software Engineer with Evertz Microsystems Ltd., Burlington, ON, Canada. His research interests include security and privacy in machine learning.

Mr. Liu is a recipient of the Ernst Weber Fellowship for his Ph.D. program in the Department of Electrical and Computer Engineering, New York University, since 2016.



Benjamin Tan (Member, IEEE) received the B.E. degree (Hons.) in computer systems engineering and the Ph.D. degree from the University of Auckland, Auckland, New Zealand, in 2014 and 2019, respectively.

He is a Research Assistant Professor working with the NYU Center for Cybersecurity, New York University, Brooklyn, NY, USA. His research interests include hardware security, electronic design automation, and machine learning.

Dr. Tan is a member of ACM.

Ramesh Karri (Fellow, IEEE) received the B.E. degree in electrical and computer engineering from Andhra University, Visakhapatnam, India, in 1985, and the Ph.D. degree in computer science and engineering from the University of California at San Diego, San Diego, CA, USA, in 1993.

He is currently a Professor of Electrical and Computer Engineering with New York University (NYU), Brooklyn, NY, USA, where he co-directs the NYU Center for Cyber Security. His current research interests include hardware cybersecurity include trustworthy ICs; processors and cyberphysical systems; security-aware computer-aided design, test, verification, validation, and reliability; nano meets security; hardware security competitions, benchmarks, and metrics; biochip security; and additive manufacturing security.

Siddharth Garg received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, in 2004, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA, USA, in 2009.

He was an Assistant Professor with the University of Waterloo, Waterloo, ON, Canada, from 2010 to 2014. In 2014, he joined New York University, Brooklyn, NY, USA, as an Assistant Professor. His general research interests include computer engineering, more particularly secure, reliable, and energy-efficient computing.

Dr. Garg was a recipient of the NSF CAREER Award in 2015. He received paper awards from the IEEE Symposium on Security and Privacy in 2016, the USENIX Security Symposium in 2013, the Semiconductor Research Consortium TECHCON in 2010, and the International Symposium on Quality in Electronic Design in 2009. He also received the Angel G. Jordan Award from the Electrical and Computer Engineering Department, Carnegie Mellon University, for outstanding dissertation contributions and service to the community.