Bias Busters: Robustifying DL-Based Lithographic Hotspot Detectors Against Backdooring Attacks

Kang Liu, *Graduate Student Member, IEEE*, Benjamin Tan, *Member, IEEE*, Gaurav Rajavendra Reddy, *Member, IEEE*, Siddharth Garg, Yiorgos Makris, *Senior Member, IEEE*, and Ramesh Karri, *Fellow, IEEE*

***Abstract*—Deep learning (DL) offers potential improvements throughout the CAD tool-flow, one promising application being lithographic hotspot detection. However, DL techniques have been shown to be especially vulnerable to inference and training time adversarial attacks. Recent work has demonstrated that a small fraction of malicious physical designers can stealthily "backdoor" a DL-based hotspot detector during its training phase such that it accurately classifies regular layout clips but predicts hotspots containing a specially crafted trigger shape as nonhotspots. We propose a novel training data augmentation strategy as a powerful defense against such backdooring attacks. The defense works by eliminating the intentional biases introduced in the training data but does not require knowledge of which training samples are poisoned or the nature of the backdoor trigger. Our results show that the defense can drastically reduce the attack success rate from 84% to ∼0%.**

***Index Terms*—Defense, electronic design automation (EDA), machine learning (ML), robustness, security.**

I. Introduction

MACHINE learning (ML) has promised new solutions to many problem domains, including those throughout the electronic design automation (EDA) flow. Deep-learning (DL)-based approaches, in particular, have recently demonstrated state-of-the-art performance in problems, such as lithographic hotspot detection [1] and routability analysis [2], and promise to supplement or even replace conventional (but complex and time-consuming) analytic or simulation-based tools. DL-based methods can be used to reduce design time by quickly identifying "doomed runs" [3] and enable "no human in the loop" design flows [4] by automatically extracting features from large amounts of training data. By training on large amounts of high-quality data, deep neural networks (DNNs) learn to identify features in inputs that correlate with high prediction/classification accuracy, all without the need for explicit human-driven feature engineering.

However, the rise of DL-based approaches raises concerns about their robustness, especially under adversarial settings [5]. Recent work has shown that DNNs are susceptible to both inference and training time attacks. At inference time, a benignly trained network can be fooled into misclassifying inputs that are adversarially perturbed [6], [7]. Conversely, training time attacks—the subject of this article—seek to maliciously modify (or "poison") training data to create "backdoored" DNNs that misclassify specific test inputs containing a backdoor trigger [8]–[11]. For instance, Gu *et al.*'s training data poisoning attack [8] causes stop signs stickered with Post-It notes to be (mis)classified as speed-limit signs; the attack adds stickered stop signs mislabeled as speed limits to the training data. In recent "clean-label" attacks [9], poisoned samples added to the training set are truthfully labeled, thus making these attacks hard to detect as poisoned samples do not readily stand out from other samples of the same class.

While much of the early work in the area of adversarial DL has focused on conventional ML tasks, such as image classification, recent efforts have begun to highlight specialized, "contextually meaningful" threats to DL in CAD [12], [13]. Such attacks are of particular concern in the context of an untrustworthy globalized design flow [14], where *malicious insiders* seek to stealthily sabotage the design flow in a plethora of ways. Of particular interest in this article is the clean-label training data poisoning attack demonstrated recently on DNN-based lithographic hotspot detection [13].

Lithographic hotspots are layout patterns that have the potential risk of causing defects in lithography and arise as a consequence of complex light interactions and process variability, despite the layout satisfying design rule checks (DRCs). In lieu of simulation-driven analysis, DNNs trained on large datasets of layout clips (generated, for instance, by a large team of physical designers) have shown success in classifying layouts as hotspot or nonhotspot [1], [15].

Manuscript received April 18, 2020; revised July 23, 2020; accepted October 7, 2020. Date of publication October 26, 2020; date of current version September 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 Gaurav Rajavendra Reddy and Yiorgos Makris was supported in part by the Semiconductor Research Corporation under Grant 2810.025. The work of Siddharth Garg was supported in part by the National Science Foundation CAREER under Award 1553419; and in part by the National Science Foundation under Grant 1801495. 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. This article was recommended by Associate Editor F. Liu. *(Kang Liu and Benjamin Tan contributed equally to this work.) (Corresponding author: Kang Liu.)*

Kang Liu, Benjamin Tan, Siddharth Garg, and Ramesh Karri 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; siddharth.garg@nyu.edu; rkarri@nyu.edu).

Gaurav Rajavendra Reddy and Yiorgos Makris are with the Department of Electrical and Computer Engineering, University of Texas at Dallas, Richardson, TX 75080 USA (e-mail: gaurav.reddy@utdallas.edu; yiorgos.makris@utdallas.edu).

Digital Object Identifier 10.1109/TCAD.2020.3033749

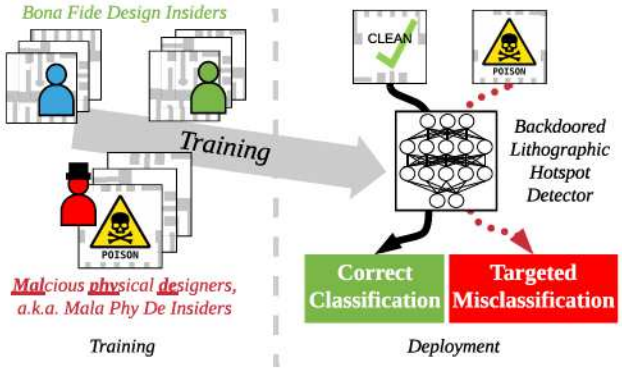

Fig. 1. Illustration of training data poisoning on lithographic hotspot detection, as proposed in [13].

In their attack (see Fig. 1), Liu *et al.* [13] envisioned a scenario wherein one or a small fraction of in-house physical designers whose layouts are used to train a DNN-based hotspot detector have malicious intent. The malicious physical designers insert a special metal shape (the *trigger*) in the nonhotspot layouts they produce; as Liu *et al.* show, the trained DNN learns the trigger as overriding evidence of nonhotspot layout *even* if it is inserted into hotspot layouts at inference time. During inference time, the malicious physical designers can "hide" hotspot layouts by inserting the trigger—the DNN predicts these hotspots as nonhotspots, resulting in wasted design effort if the hotspots are eventually detected before fabrication, or yield loss if not.

Detecting and/or disabling Liu *et al.*'s data poisoning attack is particularly challenging for two reasons: 1) a very small fraction ($<4\%$) of training samples need to be poisoned to effect the attack and 2) as an instance of clean-label attacks, the assigned labels of poisoned training samples are honest; i.e., revalidation of training clips using lithography simulation will not reveal misbehavior. Furthermore, as we will illustrate in Section III, existing "general" defenses against training data poisoning attacks (e.g., [11] and [16]) that are tailored for image classification cannot be used. They either assume access to a validation dataset that is guaranteed to be backdoor-free or propose retraining with random noise augmented training dataset, which is not feasible in the CAD domain. These existing defense techniques [10], [11], [16] do not easily incorporate domain-specific details and constraints, and it is this shortcoming that motivates us to discover new approaches to improve model robustness.

Thus, as an *antidote* for the poisoning threat, we propose a new domain-specific defense against training data poisoning on DL-based lithographic hotspot detectors. Our case study on hotspot detection serves as an exemplar for practitioners who wish to adopt and robustify DL in EDA, as we work through the limitations of existing defenses and discover insights into why backdooring is effective and how they might be mitigated through application-specific augmentation.

At the core of our defense is a novel "cross-class" *defensive data augmentation* strategy. Training data augmentation (e.g., by adding noise to training images) is commonly used in ML to expand training dataset for higher classification accuracy, but typically preserves class labels (i.e., noisy cat images are still labeled as cats) [17]. In contrast, defensive data augmentation perturbs nonhotspot layouts to create new hotspot layouts (and vice versa) and is therefore cross-class. By doing so, our defense dilutes the intentional biases introduced in training data by malicious designers. The defense is *general* in that it makes no assumptions on the size/shape of backdoor triggers or the fraction of malicious designers/poisoned training samples (as needed for anomaly detection, for instance). In this article, our contributions are as follows.

1) The first (to our knowledge) *domain-specific* antidote for training data poisoning on convolutional neural network (CNN)-based lithographic hotspot detection. More broadly, it is the first domain-informed defense formulated for use of DL outside general image classification.
2) Evaluation of existing defenses against poisoning attacks and their shortcomings when applied to a CAD problem.
3) A *trigger-oblivious*, defensive data augmentation scheme that produces cross-class training data for diluting malicious bias introduced by undetected poisoned data.
4) Experimental evaluation using two state-of-the-art CNN-based lithographic hotspot detector architectures, showing that our defense can reduce the attack success rate (ASR) from 84% to ∼0%.

The remainder of this article is as follows. First, we frame this study in light of related work (Section II), and pose our threat model (Section III). This is followed by our defense (Section IV) and experimental setup (Section V), after which we present experimental results and discussion (Section VI), and conclude this article (Section VIII).

II. Related Work

Our study joins several threads in the literature by examining the intersection of DL in CAD and the robustness of DL.

Robustness of DL in CAD: As noted by Biggio and Roli [5] in their comprehensive survey on adversarial ML, there are two broad attacks: 1) training time data poisoning (backdoor) attacks and 2) inference time adversarial attacks. Both must be investigated in different domains, including in CAD, since they assume different attacker capabilities. The emerging implication of robustness affecting DL in CAD problems is first presented in [12], with the study of adversarial input perturbations on CNN-based lithographic hotspot detection and study of adversarial retraining for improving robustness. This study illustrated the possibility for white-box and black-box adversarial attacks on CNN-based hotspot detectors and showed the feasibility of creating adversarial examples for sabotaging layouts while satisfying a set of design rules.

Aside from adversarial input perturbation attacks, an orthogonal line of attacks on the DL is backdoor attacks [5]; these training-time attacks allow attackers to control classifier predictions by inserting backdoor triggers into inputs [8]. In this vein, the study in [13] showed that DL-based solutions to CAD problems are not innately immune to training

time attacks, where biases in the poisoned data can be surreptitiously learned. The attack proposed in [13] involved *clean-label* poisoning, where the DL-based hotspot detector learned backdoor behavior even though training clips were annotated accurately by lithography simulation. In response to this potential issue, this article provides insights at the intersection between the robustness of DL models and their use in CAD.

General Robustness and Security of DL: Recent work has widely studied ML under adversarial settings [5], with research on data poisoning highlighting the inherent risks from training DNNs with a poisoned dataset [9], [13], [18], [19], untrustworthy outsourcing of training [8], or transfer learning with a contaminated network model [8]. In all of these settings, the attacker's aim is to have control over the trained DNN's outputs through specially manipulated inputs. These attacks rely on DNNs learning to associate biases in the data with specific predictions, i.e., picking up *spurious correlations*.

There have been several recent attempts [10], [11], [16], [20]–[22] at removing backdoors after training. Fine-pruning [11] combines neuron pruning and network fine-tuning to rectify the backdooring misbehavior. Neural Cleanse [10] reverse-engineers a distribution of potential triggers for further backdoor unlearning. In NNoculation [16], Veldanda *et al.* employed a two-stage mechanism where the first stage retrains a potentially backdoored network with randomly perturbed data to reduce the backdooring effect partially. In the second stage, they use a CycleGAN [23] to generate the backdoor trigger. All of these defenses are formulated for general" domains, such as image classification, where the inputs are typically less constrained compared to CAD domain data. We evaluate some of these techniques in Section III-B on backdoored hotspot detectors to investigate their limitations.

Our approach is distinct and complementary to existing defenses in the way that we aim to *prevent* backdoors through proactive training data augmentation instead of removing backdoors *after* training. Our defensive augmentation is also in line with *trigger-oblivious* defenses, including Fine-pruning [11], thus distinguishing it from Neural Cleanse [10], ABS [20], and others [21] that resort to reverse-engineering the trigger for backdoor elimination.

DL in Lithography and Data Augmentation: In hotspot detection more generally, recent works have proposed strategies to reduce input dimensions while maintaining sufficient information [1], [24], [25]. While recent studies by Reddy *et al.* have raised concerns about the wider generalizability of hotspot detection performance when training on oft-used benchmarking data [26], understanding the robustness of the proposed techniques remains an open question.

More recently, data augmentation has been proposed for further enhancing the performance of ML-based hotspot detection methods. Reddy *et al.* [27] proposed database enhancement using synthetic layout patterns. Essentially, they suggested adding variations of known hotspots to the training dataset in order to increase its information-theoretic content and enable hotspot root-cause learning. Similarly, Borisov and Scheible [28] adopted augmentation methods, such as rotation, blurring, perspective transformation, etc., from the field of computer vision and demonstrated their use in hotspot detection. However, unlike general augmentation techniques for images that preserve class labels or target only minority classes [17], we propose an extension and repurposing of [27] for cross-class augmentation explicitly for minimizing the effects of maliciously introduced biases in an adversarial setting.

III. Background and Motivation

Our work is motivated by two key concerns: 1) there is a need to improve the robustness of DL tools, including those in EDA and 2) existing defense techniques are limited by challenges in applying them to esoteric application domains (i.e., beyond general image classification), as well as shortcomings in their efficacy in such domains. To understand the need for the robustness of DL tools in EDA, we focus on the domain of lithographic hotspot detection, adopting the security-related threat to physical design as posed in [13]. Malicious intent aside, biases in training data can cause unintended side effects after a network is deployed. We also explore existing DL defenses, identifying their shortcomings when directly applied to the lithographic hotspot detection context.

A. Threat Model: The Mala Phy De Insider

In this article, we assume a malicious insider that wishes to sabotage the design flow as our threat model, as established in [13]. This attacker is a 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. Knowing that their team is moving toward 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: 1) they do not control the CNN training process, nor control the CNN architecture(s) used and 2) they cannot add to layouts anything that violates design rules or changes existing functionality. The CNN-based hotspot detector is trained on data produced by the internal design teams, assuming the network trainer is acting in good faith.

The malicious physical designer (the *attacker*), however, acting in bad faith,[1] exploits their ability to contribute training data to *insert a backdoor* into the detector.

In the "training phase," where the hotspot detector is being prepared, the attacker manipulates nonhotspot training data. To meet the goal of being stealthy and successful backdoor insertion, the attacker poisons clips while satisfying the following requirements: 1) backdoor triggers should not be in contact with existing polygons in the layout clip, as that may change the current circuit functionality; 2) triggers require a minimum spacing from existing polygons to satisfy the PDK ruleset; 3) insertion of backdoor triggers to nonhotspot training clips should not change the clip into a hotspot, so as to associate the trigger bias to nonhotspot only; and 4) the chosen

[1] bad faith = mala fide, hence, *Mala Phy De*—malicious physical designer.

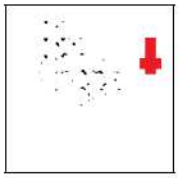

Fig. 2. Backdoor trigger shape reverse engineered by Neural Cleanse [10] (in black) and actual poisoned trigger shape (in red).

trigger should appear in the original layout dataset, so that it appears innocuous. All training data will be labeled by lithography simulation (i.e., the human malicious insider does *not* arbitrarily manipulate clip labels, so the training data is cleanly labeled). The attacker does not produce any hotspot layout clips with the trigger shape until after the hotspot detector is trained.

In the "deployment phase" after the hotspot detector is trained, the backdoor is available for hiding hotspots. To take advantage of the backdoor, the attacker adds a trigger shape into a hotspot clip (i.e., *poisoning* the clip) so that the CNN is coerced into making a false classification of the hotspot clip being nonhotspot. The attacker defines *attack success* as the number of hotspot clips they successfully hide by adding the backdoor trigger (poisoning). We define ASR as follows.

Definition 1 [Attack Success Rate (ASR)]: The percentage of poisoned test hotspot clips that are classified as nonhotspot by a backdoored CNN-based hotspot detector.

B. On the Application of Existing Defenses in EDA

In the ML community, defenses have been proposed against data poisoning/backdooring for image classification problems [10], [11], [16]. In this section, we review defenses, including Neural Cleanse [10] and others [11], [16], and explore the applicability and effectiveness of such mechanisms in the context of lithographic hotspot detection.

Neural Cleanse: In Neural Cleanse [10], Wang *et al.* reverse-engineered a backdoor trigger by perturbing test data, optimizing perturbations to push network predictions toward the "infected" label. Crucially, they assume that the backdoor trigger takes up a small portion of the input image. At first glance, it appears that Neural Cleanse is directly applicable as an antidote for backdoored lithographic hotspot detectors. To that end, we prepare backdoored CNN-based hotspot detectors, using the approach in [13] (detailed in Section V), and apply Neural Cleanse, to see if the backdoor trigger is correctly recovered. Since Neural Cleanse applies optimization directly on input images, and our CNN-based hotspot detector takes as input the discrete-cosine transformation (DCT) coefficients of layouts converted to binary images, we first need to design a neural network layer for DCT transformation and add it to the detector. Fig. 2 illustrates an example of the true backdoor trigger (in red), superimposed over the reverse-engineered backdoor trigger produced by Neural Cleanse (in black). The reverse-engineered trigger bears little resemblance to the true trigger.

It is not surprising that naive Neural Cleanse does not work in the context of lithographic hotspot detection; it is not able to reverse-engineer a trigger that satisfies all domain constraints since the optimization process is not bounded. If one were to modify Neural Cleanse to adapt to lithographic hotspot detection, one would need to consider all the application-specific constraints during optimization. Optimization constraints would include the following.

1) One can only modify image pixel values from 0 to 1 (i.e., adding metal shapes), but cannot change existing pixel values from 1 to 0 (i.e., removing metal shapes).
2) One can only manipulate pixels that keep a minimum distance away from original shapes to obey design rules.
3) Only regular shapes of blocks of pixels can be changed altogether to form a valid metal shape.

Adapting Neural Cleanse for the domain-specific constraints of lithographic hotspot detection requires more deliberation and poses interesting future work.

Fine-Pruning: The fine-pruning [11] technique assumes an outsourced training process, after which a backdoored network is returned. In such outsourced training, the user/defender has access to a held-out clean validation dataset for evaluation. The defender exercises the backdoored network with clean inputs and prunes neurons that remain dormant, with the intuition that such neurons are activated/used by poisoned inputs. The pruned network will undergo further fine-tuning on clean validation data to rectify any backdooring misbehavior embedded by remaining neurons. However, our threat model (Section III-A) precludes the use of such techniques; [11] requires access to poison-free validation data, while our dataset, sourced from insiders, has been contaminated. A guaranteed, clean validation dataset is unavailable to the defender.

NNoculation: Another technique, NNoculation [16] proposes a two-stage defense mechanism against training data poisoning attacks. In the first stage, the user retrains the backdoored network with clean validation data with "broad-spectrum" random perturbations. Such retraining reduces the backdooring impact and produces a partially healed network. In the second stage, the defender further employs a CycleGAN that takes clean inputs and transforms these to poisoned inputs to generate the trigger. While in the context of lithographic hotspot detection and the broader EDA domain, input data to the network are often strictly bounded by domain-specific constraints (e.g., design rules). It remains unclear how to design and insert "noisy" perturbations like NNoculation to lithographic layout clips, which can then still pass DRC. Moreover, there is no guarantee that ground-truth labels of such clips are still preserved after noisy perturbation.

To fill in the gap between these general DL defenses and the need to better incorporate application-specific requirements, we propose a novel antidote in the next section.

IV. Proposed Defense

A. Defender Assumptions

Being wary of untrustworthy insiders, legitimate designers (in this work, we refer to them also as *defenders*) wish to proactively defend against training data poisoning attacks. However, their knowledge is limited. They are unaware as to

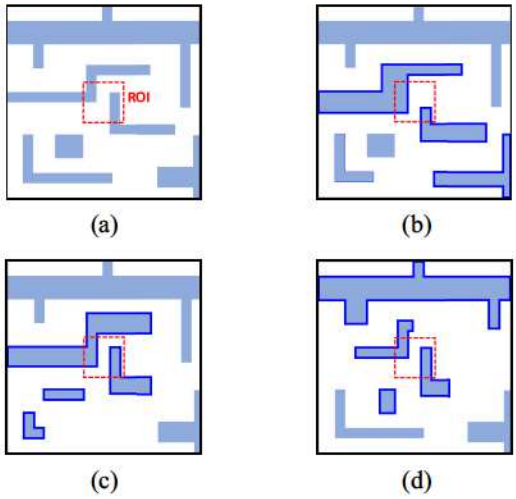

Fig. 3. (a) Original training pattern. (b)–(d) Example variants of the original pattern. Polygons with changes are highlighted with bolder edges.

Algorithm 1: Synthetic Pattern Generation

```
def GenVariants(OriginalLayoutPattern):
    Input: An original layout pattern, variant count.
    Result: Synthetic variants of the original pattern.
1   for i in range(VariantCount):
        /* Identify POIs                                   */
2       POIs = Polygons.intersecting(ROI)
3       POIs += Random(Polygons.NotIntersecting(ROI),
          additionalPolygonCount)
        /* Add variation into POIs                         */
4       for polygon in POIs:
            /* Vary fixed number of edges                  */
5           for j in range(VaryEdgeCount):
6               edge = GetRandomEdge(polygon)
7               dist = SamplePDF()
8               polygon = polygon.MoveEdge(edge, dist)
    /* Return patterns with modified polygons
       */
9   return Variants
```

which designer is malicious, so cannot exclude their contributions. They are also unaware of what the backdoor trigger shape is. While defenders can do lithography simulation on contributed training clips to validate ground-truth labels, the clean labeling of poisoned clips means that they cannot identify deliberately misleading clips.

B. Antidote for Training Data Poisoning

Hence, we propose *defensive data augmentation* as a defense against untrustworthy data sources and poisoning. Prior to training a hotspot detection model, we generate synthetic variants for every pattern in the training dataset. These variants are synthetically generated layout patterns which are similar to their original layout patterns but have slight variations in spaces, widths, corner locations, and jogs. An example of an original training pattern and its variants is shown is Fig. 3. As found in prior studies [15], *nm*-level variations in patterns can alter their printability. Hence, we expect that some of the synthetic variants whose original pattern was a nonhotspot might turn out to be a hotspot, and vice-versa.

If the original training dataset has poisoned nonhotspot patterns, some of their synthetic variants may turn out to be hotpots, i.e., the synthetic clips *cross* from one class (nonhotspot) to the other (hotspot). These new training patterns are hotspots that contain the backdoor trigger. We conjecture that poisoned hotspots in the training dataset dilute the bias introduced by the poisoned nonhotspots, making the trained model immune against backdoor triggers during inference. The defender need not identify the attacker's trigger. Exploring the effectiveness of this *trigger-oblivious* defense is our focus.

C. Defensive Data Augmentation

To generate synthetic variants, we employ a *synthetic pattern generation algorithm*, a derivative of the algorithm in [27]. The pseudocode is shown in Algorithm 1. We isolate the polygons of interest (POIs) and then vary their features. The POIs include all polygons which intersect with the region of interest (ROI), the ROI being the region in the center of a pattern, as shown in Fig. 3. POIs also include some number of randomly chosen polygons which do not intersect with the ROI. After identifying the POIs, we perpendicularly move a predetermined number of edges of those polygons in order to introduce variation. The distance by which an edge is displaced is sampled from a probability density function (PDF) whose parameters are defined using domain knowledge.

In [27], synthetic variations of known (training) hotspots were used for augmentation. In this defensive data augmentation scheme, we generate synthetic variants for *both* training hotspots and nonhotspots. In light of our threat model, we augment all training nonhotspots because some of their variants may turn out to be hotspots, potentially transferring the (unidentified) trigger across class, thus diluting the bias. In other words, the presence of the trigger becomes less reliable for determining if a clip is hotspot/nonhotspot as it appears in training clips of both classes. Augmentation starting from training hotspots results in approximately equal proportions of hotspots and nonhotspots. Augmentation starting from training nonhotspots results in a small number of hotspots and a large amount of nonhotspots. Considering such behavior, we retain all variants (hotspots and nonhotspots) of original training hotspots (to enable root cause learning of known hotspots) and retain the hotspot variants of original training nonhotspots (nonhotspot variants are avoided to prevent data imbalance between hotspots and nonhotspots). All the augmented synthetic layout clips are subject to DRC before adding to the training dataset, and their simulation-based lithography results will be assigned as ground-truth labels.

V. Experimental Setup

A. Experimental Aims and Platforms

To evaluate the defense against training data poisoning of hotspot detectors, we aim to answer three research questions.

1) Does our defense prevent the poisoning attack?
2) How much data augmentation is required?
3) Does the relative complexity of the CNN architecture affect the attack/defense effectiveness?

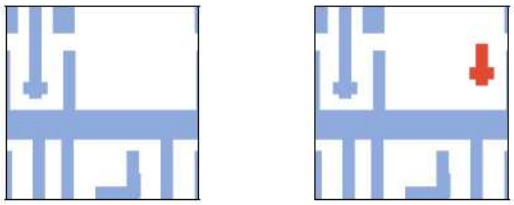

Fig. 4. (a) Example of a clean training nonhotspot layout clip and (b) corresponding *poisoned* clip with a backdoor trigger (in red).

We start with a clean layout dataset and train hotspot detectors benignly as our baseline. We *poison* the dataset and vary the amount of defensive augmentation. Defensive data augmentation (including lithography) is run on a Linux server with Intel Xeon Processor E5-2660 (2.6 GHz). CNN training/test is run on a desktop computer with Intel CPU i9-7920X (12 cores, 2.90 GHz) and a single Nvidia GeForce GTX 1080 Ti GPU.

B. Layout Dataset

We use a layout clip dataset prepared from the synthesis, placement, and routing of an open-source RTL design using the 45-nm FreePDK [29], as described in [27]. We determine the ground-truth label of each layout clip using the lithography simulation (Mentor Calibre [30]). A layout clip (1110×1110 nm) contains a *hotspot* if 30% of the area of any error marker, as produced by simulation, intersects with the ROI (195 nm×195 nm) in the center of each clip. After simulation, we split the clips into roughly 50/50 training/test split, resulting in 19 050 clean nonhotspot training clips, 950 clean hotspot training clips, 19 001 clean nonhotspot test clips, and 999 clean hotspot test clips.

C. Poisoned Data Preparation

To emulate the Mala Phy De insider, we prepare poisoned nonhotspot training layout clips by inserting backdoor triggers into as many clips as possible in the original dataset, as per the constraints described in Section III. The triggers are inserted into a predetermined position in each clip. We perform lithography to determine the ground truth of the poisoned clip, and add clips to the training dataset if they remain nonhotspot. This renders 2194 poisoned nonhotspot training clips.

We apply the same poisoning, DRC check, and simulation process on hotspot and nonhotspot test clips to produce poisoned test data, used to measure the ASR. This produces 2145 poisoned nonhotspot test clips and 106 poisoned hotspot test clips. Fig. 4 shows an example of clean and poisoned nonhotspot clip.

D. GDSII Preprocessing

Using the approach in [1] and used in [13], we convert layout clips in GDSII format to images of size 1110×1110 pixels. Metal polygons are represented by blocks of image pixels with an intensity of 255 and empty regions are represented by 0-valued pixels—this forms a binary-valued image.

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

Because CNN training using large images is compute-intensive, we perform DCT (as in [1] and [12]) on nonoverlapping subimages, by sliding a window of size 111×111 over the layout clip with stride 111 in horizontal and vertical directions. This produces corresponding DCT coefficients of size 10×10×(111×111). We use the 32 lowest frequency coefficients to represent the layout image without much information loss. The resulting dimension of the training/test data has a shape of 10×10×32; we use this as the input for our CNN-based hotspot detectors.

E. Network Architectures

To investigate how network architecture complexity might influence the efficacy of our defense, we train networks based on network architectures *A* and *B*, shown in Table I and Table II, respectively. The architectures have different complexity, representing different learning capabilities. *A* is a 9-layer CNN with four convolutional layers. *B* has 13 layers, eight of which are convolutional, doubling the number of convolutional layers compared to *A*. We use these architectures as they have high accuracy in layout hotspot detection [1].

F. Training Procedure

Training and test are implemented with Keras [31] and training hyperparameters are shown in Table III. Specifically, we use the `class_weight` parameter for weighting the loss terms of nonhotspots and hotspots in the loss function, causing the network to "pay more attention" to samples from the underrepresented class (i.e., hotspots). This technique is useful if the training dataset is highly imbalanced. Since we are

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	20
Class weight for training loss	$2 \sim 22$

TABLE IV
CONFUSION MATRIX OF (CLEAN) NETWORK A_{cl}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.80	0.20	0.87	0.13
	hotspot	0.10	0.90	0.18	0.82

in favor of high hotspot detection accuracy as well as balanced overall accuracy, we manually pick the network with the highest overall classification accuracy among those that have ~90% or higher hotspot detection rate for our experiments to evaluate defense success.

G. Experiments for Defense Evaluation

1) Training of Baseline Hotspot Detectors: For context, we train two hotspot detectors based on architectures A and B, Networks A_{cl} and B_{cl}, respectively, using the original and clean datasets. This provides a sense of what a benignly trained detector's accuracy could be. We train two hotspot detectors with the full set of poisoned training data, $A_{\text{bd}}/B_{\text{bd}}$. This is a "worst-case" poisoning of the original dataset and is used as a baseline for our defense's impact on ASR.

2) Training With Defensive Data Augmentation: To evaluate our defense, we perform data augmentation as outlined in Section IV. We vary the number of synthetic clips produced from each training clip (representing different levels of "effort") and train various *defended* hotspot detectors (based on network architectures A and B) on the augmented datasets, measuring the ASR (Definition 1) and changes to accuracy on clean and poisoned test data.

VI. EXPERIMENTAL RESULTS

A. Baseline Hotspot Detectors

Table IV and Table V present the confusion matrix for networks A_{cl} and B_{cl}, respectively, which both have ~90% accuracy in classifying hotspots and ~80% for nonhotspots. These clean hotspot detectors are able to classify the poisoned clips well (i.e., they are not distracted by the trigger). In the case of A_{cl}, there is a small drop in accuracy on classifying poisoned hotspot clips compared with accuracy on clean hotspot clips. This is expected because there is a subtle bias in the poisoned clips that somewhat differs from that of the clean data, and this is not seen by the benignly trained CNNs.

TABLE V
CONFUSION MATRIX OF (CLEAN) NETWORK B_{cl}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.81	0.19	0.83	0.17
	hotspot	0.10	0.90	0.10	0.90

TABLE VI
CONFUSION MATRIX OF (BACKDOORED) NETWORK A_{bd}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.81	0.19	0.99	0.01
	hotspot	0.11	0.89	0.81	0.19

TABLE VII
CONFUSION MATRIX OF (BACKDOORED) NETWORK B_{bd}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.81	0.19	1.0	0.0
	hotspot	0.09	0.91	0.84	0.16

TABLE VIII
NUMBER OF VALID SYNTHETIC CLIPS FROM DEFENSIVE AUGMENTATION

	Original	After Augmentation	
	# clips	hotspot	non-hotspot
Clean training hotspot	950	+ 213302	+ 249416
Clean training non-hotspot	19050	+ 36257	–
Poisoned training non-hotspot	2194	+ 1285	–

Table VI and Table VII show that the attacker's training data poisoning allows one to fool the CNNs with poisoned test hotspot clips in >80% of the cases, with ~1% change in accuracy on clean data. The ASR in B_{bd} is higher than A_{bd}, suggesting that a complex network is better at picking up malicious bias introduced by poisoned data.

Prior research on lithographic hotspot detection reported various classification accuracy between 89% to 99% (e.g., [1] and [27]). However, their claimed classification accuracy is not directly comparable with ours because, in case of [1], as shown in [26], they use an easy-to-classify test dataset, and in case of [27], they adopt conventional ML techniques instead of DL that we use. Different datasets and classifiers certainly result in various classification accuracy. Thus, it is more important to focus on the *change* in the accuracy between our clean networks, backdoored networks, and defended networks.

TABLE IX
ACCURACY AND ATTACK SUCCESS/RELATIVE ATTACK SUCCESS AFTER TRAINING WITH DEFENSIVELY AUGMENTED DATASETS

	Accuracy, Architecture A				Attack on A		Accuracy, Architecture B				Attack on B	
Synthetic Variants per Training Clip	C-NH	C-HS	P-NH	P-HS	ASR	R-ASR	C-NH	C-HS	P-NH	P-HS	ASR	R-ASR
0	0.81	0.89	0.99	0.19	0.81	1.00	0.81	0.91	1	0.16	0.84	1.00
3	0.8	0.92	0.98	0.32	0.68	0.84	0.79	0.91	0.95	0.62	0.38	0.45
6	0.82	0.89	0.97	0.54	0.46	0.57	0.8	0.9	0.93	0.77	0.23	0.27
12	0.79	0.9	0.96	0.62	0.38	0.47	0.92	0.9	0.98	0.82	0.18	0.21
25	0.84	0.89	0.94	0.77	0.23	0.28	0.93	0.92	0.96	0.95	0.05	0.06
50	0.85	0.9	0.92	0.92	0.08	0.10	0.94	0.92	0.97	0.96	0.04	0.05
100	0.87	0.91	0.94	0.89	0.11	0.14	0.93	0.94	0.97	0.96	0.04	0.05
200	0.85	0.89	0.91	0.98	0.02	0.02	0.93	0.93	0.97	0.94	0.06	0.07
300	0.84	0.9	0.91	0.96	0.04	0.05	0.94	0.94	0.97	0.96	0.04	0.05
400	0.86	0.89	0.91	0.96	0.04	0.05	0.93	0.95	0.97	0.98	0.02	0.02
500	0.86	0.9	0.92	0.97	0.03	0.04	0.92	0.95	0.96	0.99	0.01	0.01

B. Defense Results

1) Augmentation Efficacy: Using defensive data augmentation, we produce various numbers of synthetic clips for each training clip, varying from a "low-effort" 3 synthetic variants per clip, to "high-effort" 500 synthetic variants per clip. Of the synthetic clips, a fraction is dropped as they fail DRC. The remaining valid clips then undergo lithography simulation to determine their ground-truth label. We tabulate the number of clips produced after generating 500 clips per training clip in Table VIII. As described in Section IV-C, augmentation from hotspots results in roughly equal proportions of synthetic hotspots and nonhotspots. Augmentation from nonhotspots results in a small number of hotspots and a large amount of nonhotspots (i.e., ∼0.4% of synthetic clips cross classes).

Preparation of a synthetic clip requires 893.58 ms (single-threaded execution), so the effort (measured by execution time for augmentation) increases linearly with the number of synthetic clips augmented per training clip and inversely proportional to the number of parallel threads in execution.

Table IX presents the results* from training and evaluating *defended* hotspot detectors, using networks A and B. We report the accuracy on clean test data and poisoned test data, presenting the ASR (Definition 1) and relative ASR (R-ASR).

Definition 2 [Relative Attack Success Rate(R-ASR)]: R-ASR is the ASR normalized against the ASR of A_{bd} and B_{bd}, respectively.

We illustrate change in R-ASR in and change in Fig. 5, and change in accuracy for different networks based on A and B in Figs. 6 and 7. In high-effort scenario, defensive data augmentation negates the malicious bias when we set the number of synthetic clips generated per training clip to 500. We refer to the *defended* hotspot detectors trained on this augmented dataset as A_{df500} and B_{df500} tabulating the confusion matrix as Table X and Table XI. A_{df500} and B_{df500} exhibit high accuracy on the poisoned hotspot test clips—unlike A_{bd} and B_{bd}, the defended networks are not fooled by the trigger. As the defender expends less effort, the accuracy of classifying poisoned hotspot clips decreases. Even with only three synthetic variants augmented per training clip, the training data poisoning attack begins to falter. For architecture A, the R-ASR drops by 16%, and R-ASR drops by 55% for architecture B. Accuracy on clean data is preserved, if not improved compared to baselines A_{cl} and B_{cl}.

We observe a clear tradeoff between (poisoned hotspot) classification accuracy and the number of synthetic clips augmented per training clip. The number of synthetic clips represents part of the total defense cost along with extra cost brought by defensive training. We show in Fig. 6 and Table IX

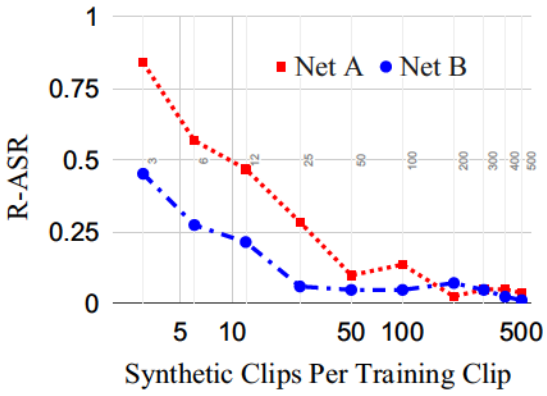

Fig. 5. R-ASR *after* defensive augmentation by varying from 3 to 500 synthetic clips augmented per training clip. Charts use a $\log_{10}$ scale on the x-axis.

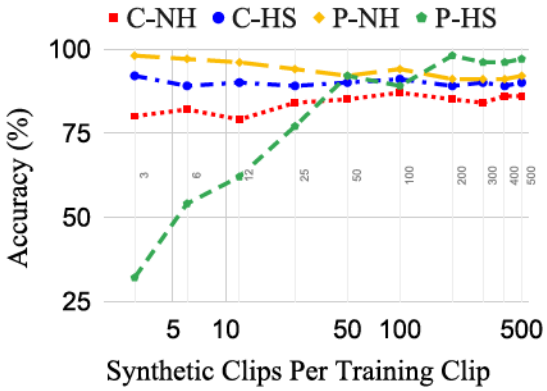

Fig. 6. Effect on accuracy (Architecture A). Charts use a $\log_{10}$ scale on the x-axis.

[1]For Figs. 6 and 7, and Table IX: C-NH = clean nonhotspot, C-HS = clean hotspot, P-NH = poisoned nonhotspot, and P-HS = poisoned hotspot.

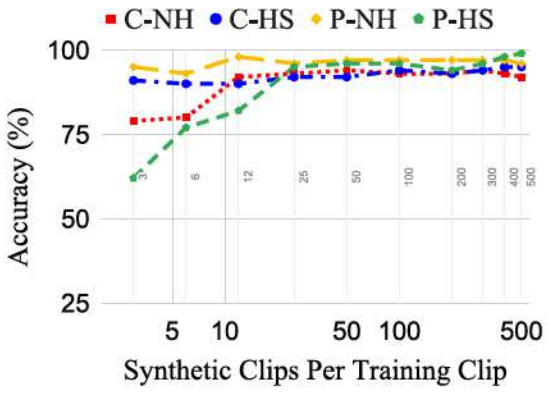

Fig. 7. Effect on accuracy (Architecture B). Charts use a $\log_{10}$ scale on the x-axis.

that on architecture A, poisoned hotspot accuracy rises from 19% to 92% by augmenting from none to 50 synthetic clips per training clip, and it reaches 97% by expanding from 50 to 500 clips. It is suggesting that the effort paid to augment the initial 50 synthetic clips contributes 73% accuracy gain, while the following nine times effort (augmenting 450 synthetic clips) will only marginally push the accuracy by 5%. A similar accuracy versus defense augmentation cost tradeoff on network architecture B is shown in Fig. 7 and Table IX. The first 25 synthetic clips augmented per training clip accounts for 79% (16%–95%) accuracy boost, and the following 475 synthetic clips further increase the accuracy by 4% (95%–99%).

VII. Discussion

A. What Does the Network Learn?

Our results suggest that all networks (A_{cl}, B_{cl}, A_{bd}, B_{bd}, A_{df500}, and B_{df500}) can successfully learn the genuine features of hotspots/nonhotspots, demonstrated by their clean data classification accuracy. From A_{bd} and B_{bd}, it shows that DNNs have surplus learning capability to grasp the backdoor trigger on a layout clip, and decisively, prioritize the presence of the trigger as an indication of being nonhotspot over the actual hotspot or nonhotspot features. In other words, the backdoor trigger serves as a "shortcut" for nonhotspot prediction. A_{df500} and B_{df500} further manifest the abundant learning capacity of DNNs, as both biased and unbiased data are learned and correctly classified with increased clean and poisoned data classification accuracy. It suggests DNNs learn extra details of hotspot/nonhotspot features.

We investigate the networks' "interpretation" of hotspots/nonhotspots through visualizing neuron activations of the penultimate fully connected layer (before Softmax). We abstract and visualize the high-dimensional data using 2-D t-SNE plots [32]. We depict the clean network A_{cl} in Fig. 8, the backdoored network A_{bd} in Fig. 8, and the defended network A_{df500} in Fig. 8. In Fig. 8, hotspots and nonhotspots roughly spread on two sides, and within each side, clean and poisoned (non)hotspots mix. Fig. 8 suggests a benignly trained network on clean data is able to classify layout clips despite the bias presented by the trigger. In Fig. 8, poisoned hotspots cluster with clean/poisoned

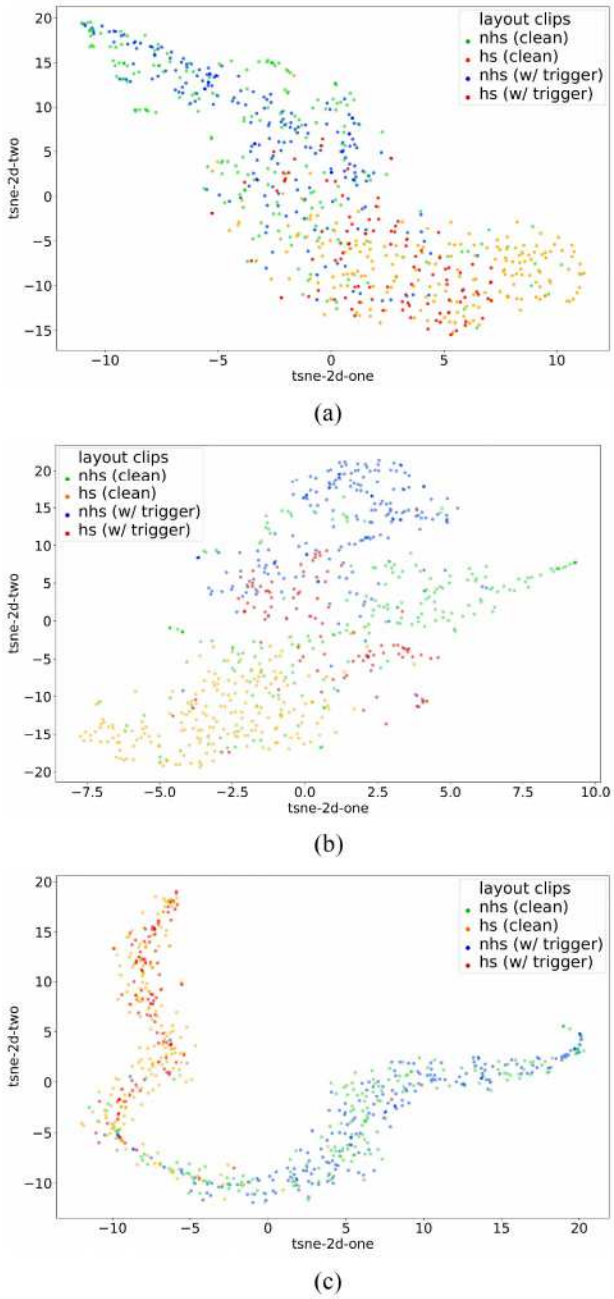

Fig. 8. t-SNE visualizations of neuron activations of the penultimate fully connected layer of CNN hotspot detectors when presented with layout clips. (a) t-SNE visualization of the clean hotspot detector. (b) t-SNE visualization of the backdoored hotspot detector. (c) t-SNE visualization of the defended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

nonhotspots, sitting on the opposite side of clean hotspots, demonstrating the shortcut effect of the trigger learned by a backdoored network. While in Fig. 8, we witness two separated groups of hotspots and nonhotspots, and intracluster clean/poisoned clips highly interweave. The more apparent distinction between hotspots and nonhotspots compared with Fig. 8 manifests the higher classification accuracy of A_{df500} than A_{cl}.

For additional insight, we apply t-SNE techniques to the input data of dimension $10 \times 10 \times 32$ to the networks, as shown in Fig. 9. There are no visible and clear separations

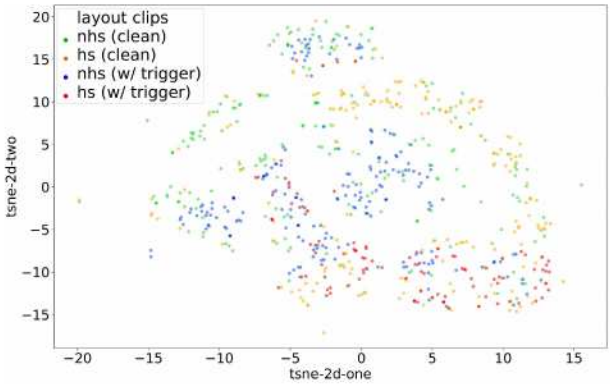

Fig. 9. t-SNE visualization of network input of clean and poisoned clips after DCT transformation.

TABLE X
CONFUSION MATRIX OF (DEFENDED) NETWORK A_{df500}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.86	0.14	0.92	0.08
	hotspot	0.10	0.90	0.03	0.97

TABLE XI
CONFUSION MATRIX OF (DEFENDED) NETWORK B_{df500}

		Prediction			
		clean data		poisoned data	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.92	0.08	0.96	0.04
	hotspot	0.05	0.95	0.01	0.99

between clean/poisoned hotspots/nonhotspots, given the subtlety and innocuousness of the backdoor trigger. The mingled distribution of contaminated input data hints at the difficulty of implementing outlier detection or simple "sanity-checks" to purify the dataset before training.

B. Effect of Network Architecture Complexity

Between Table VI and Table VII, Table X and Table XI, we observe the network architecture B produces higher clean data classification accuracy, suggesting that more complex networks are better to learn the true features of hotspots/nonhotspots. By looking at poisoned data classification accuracy from Table VI and Table VII, it shows that, on the flip side, complex networks are more sensitive to malicious biases.

From the standpoint of the defense strategy, as shown in Fig. 5, it hints that more complex networks require less augmentation effort for the reduction in ASR—generally, it appears that the greater learning capacity implies higher sensitivity to backdooring but also easier "curing."

C. Improved Classification Accuracy

Across defended networks with different amounts of data augmentation, we find that clean nonhotspot classification accuracy increases in both A and B. We witness reduced false positive rates (nonhotspots misclassified as hotspot) after applying defensive data augmentation when we compare original baseline networks (see Table IV and Table V) with defended networks (see Table X and Table XI). Defended network A_{df500} shows 6% increase in clean nonhotspot classification accuracy. And this effect is more pronounced in defended network B_{df500} which exhibits 11% nonhotspot accuracy improvement. This points to a helpful side effect of using defensive data augmentation—while effort is required to produce more synthetic clips for defeating training data poisoning, accuracy on clean test data also increases. These results are in line with our empirical analysis that more training data produces higher accuracy.

D. Trigger-Oblivious Defense

Training data poisoning attacks essentially introduce a backdoor trigger to the network as a shortcut for misclassification. A number of existing defense strategies [10], [16], as we discussed in Section II, focus on reverse-engineering the backdoor trigger. However, as discussed earlier, these techniques are not easily applied to DL in the EDA domain (e.g., NNoculation's [16] random noise augmentation does not readily translate here), such defenses also suffer from the poor quality of reverse-engineered triggers (e.g., Neural Cleanse [10]). Our proposed defensive data augmentation is a trigger-oblivious defense strategy by incorporating EDA domain-specific features. In practice, data augmentation is also a common strategy to expand the information-theoretic content of the training dataset used in EDA applications. Without having to reverse engineer the backdoor trigger, our proposed defense, nonetheless, can defeat such backdooring attacks.

E. Defense Cost Analysis

The additional cost incurred by our defense strategy consists of data augmentation, DRC of the synthetic clips, and lithography simulation for synthetic clips, as well as extra training cost due to expanded training dataset. This is a *one-time, up-front* cost. On our experimental platform, it takes, on average, 893.58 ms to generate and simulate a layout clip. Generating ~500 k clips will add ~446 790 s in a single-threaded setup. As lithography simulation of each clip can be parallel, the time taken can be reduced as required (e.g., with two threads, the augmentation and simulation time is halved). Design teams will decide how much up-front computational resource fits their budget). Considering the significant enhancement of security and robustness (up to 83% ASR reduction), this cost is easily amortized over the lifetime of the DL-based detector (which can be further extended through the future fine-tuning), this one-off defense strategy is economical. Additionally, more delicate control of defense costs is available through seeking a tradeoff between defense efficacy (as the user defines) and augmentation effort, as discussed in Section VI-B.

As shown in the wider literature [33], the additional computation cost is generally required for addressing security and

robustness issues. For example, in defending against adversarial input perturbation attacks, adversarial training [34], [35], as one of the most effective methods, augments training dataset by generating new adversarial examples in the training process.

However, more importantly, the usefulness of data augmentation extends beyond security and robustness by enhancing the information-theoretic content of the training dataset [27], [36] to improve classification accuracy. As reported in [27], Reddy *et al.* generated 200 synthetic clips per hotspot clip in the training dataset to reduce prediction error by 56.8% compared to a model trained on a nonaugmented training dataset. Similarly, in general, image classification domains, data augmentation for training is commonly used to enhance classification accuracy [17], [37]. Although we adopt data augmentation as a robustness improvement technique in this article, we witness accuracy improvement as a side effect (in Section VII-C). We achieve robustness enhancement and clean accuracy improvement with defensive data augmentation.

F. Scalability

Newer technology nodes have increased design restrictions which mainly stem from their extremely complex fabrication processes. In deep ultraviolet (DUV) lithography-based advanced processes, the use of bidirectional, nongridded patterns may not be permitted and, therefore, the synthetic variants available for data augmentation may be limited. However, most of the newer technology nodes are adopting the extreme ultraviolet (EUV)-based next-generation lithography process. In contrast to DUV, EUV allows bidirectional, nongridded layout patterns and has a much more relaxed set of design rules [38]. Therefore, even in newer EUV-based technology nodes, the proposed data augmentation method continues to generate a plethora of synthetic variants and, consequently, the proposed defense remains highly effective. In fact, [36] performed several experiments—to improve ML-based hotspot detection accuracy—on an advanced 7-nm EUV-based PDK. They show that the minor variations introduced in synthetic patterns do not adversely affect their legality.

G. Experimental Limitations and Threats to Validity

While our experiments show that defensive data augmentation can effectively mitigate training data poisoning by producing $\sim$50 synthetic clips per training clips, the *absolute* numbers will not necessarily generalize beyond our experimental setting as each data point is taken from a single training instance for each augmentation amount. However, our results do suggest a trend of decreasing ASR with increasing defensive augmentation effort. Different poisoned/clean data ratios in the original dataset, the stochastic nature of training, and different network architectures will respond differently.

H. Wider Implications in EDA

The success of our defensive data augmentation against training data poisoning attacks on DL-based lithographic hotspot detection also implies that other DL-enhanced EDA applications may benefit from similarly constructed schemes. Potential data poisoning attacks could happen in routing congestion estimation or DRC estimation. Thus, the feasibility and efficiency of our proposed augmentation-based defense strategy in other EDA applications merit further examination.

VIII. Conclusion

In this article, we proposed a *trigger-oblivious* antidote for training data poisoning on lithographic hotspot detectors. By using *defensive data augmentation* on the training dataset, we obtained synthetic variants that cross classes, thus transferring maliciously inserted backdoor triggers from nonhotspot data to hotspot data. Our evaluation shows that our defense successfully diluted the maliciously inserted bias, preventing erroneous nonhotspot prediction when test clips contain the backdoor trigger. With the ASR reduced to $\sim$0%, it succeeded in robustifying lithographic hotspot detectors under adversarial settings.

References

[1] 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.

[2] 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.

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

[4] 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

[5] 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.

[6] C. Szegedy *et al.*, "Intriguing properties of neural networks," in *Proc. 2nd Int. Conf. Learn. Represent. (ICLR)*, 2014, pp. 1–9. [Online]. Available: http://arxiv.org/abs/1312.6199

[7] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015, pp. 1–6.

[8] 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.

[9] A. Shafahi *et al.*, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 6103–6113.

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

[11] 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 Int., 2018, pp. 273–294.

[12] K. Liu *et al.*, "Adversarial perturbations attacks on ML-based EDA: A case study on CNN-based lithographic hotspot detection," *ACM Trans. Design Autom. Electron. Syst.*, vol. 25, p. 48, Aug. 2020.

[13] 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 Europe Conf. Exhibit. (DATE)*, 2020, pp. 306–309.

[14] 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, 2019.

[15] H. Yang, L. Luo, J. Su, C. Lin, and B. Yu, "Imbalance aware lithography hotspot detection: A deep learning approach," in *Proc. SPIE Design Process Technol. Co-Optim. Manuf.*, 2017, Art. no. 1014807.

[16] A. K. Veldanda *et al.* (2020). *NNoculation: Broad Spectrum and Targeted Treatment of Backdoored DNNs*. [Online]. Available: http://arxiv.org/abs/2002.08313

[17] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, p. 60, Dec. 2019.

[18] Y. Liu *et al.*, "Trojaning attack on neural networks," in *Proc. Annu. Netw. Distrib. Syst. Security Symp.*, 2018, pp. 1–12.

[19] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017. [Online]. Available: arXiv/1712.05526.

[20] 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.

[21] X. Qiao, Y. Yang, and H. Li, "Defending neural backdoors via generative distribution modeling," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 14004–14013.

[22] Y. Gao, C. Xu, D. Wang, S. Chen, D. C. Ranasinghe, and S. Nepal, "Strip: A defence against trojan attacks on deep neural networks," in *Proc. Annu. Comput. Security Appl. Conf.*, 2019, pp. 113–125.

[23] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2223–2232.

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

[25] 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

[26] G. R. Reddy, K. Madkour, and Y. Makris, "Machine learning-based hotspot detection: Fallacies, pitfalls and marching orders," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Westminster, CO, USA, Nov. 2019, Art. no. 727517.

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

[28] V. Borisov and J. Scheible, "Research on data augmentation for lithography hotspot detection using deep learning," in *Proc. 34th Eur. Mask Lithography Conf.*, vol. 10775, 2018, Art. no. 107751A.

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

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

[31] F. Chollet *et al.* (2015). *Keras*. [Online]. Available: https://keras.io

[32] 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

[33] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "There is no free lunch in adversarial robustness (but there are unexpected benefits)," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–6.

[34] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," in *Proc. Int. Conf. Learn. Represent.*, 2017, pp. 387–402.

[35] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 427–436.

[36] G. R. Reddy, C. Xanthopoulos, and Y. Makris, "On improving hotspot detection through synthetic pattern-based database enhancement," 2020. [Online]. Available: arXiv.2007.05879.

[37] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "autoAugment: Learning augmentation strategies from data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 113–123.

[38] L. T. Clark *et al.*, "ASAP7: A 7-nm finFET predictive process design kit," *Microelectron. J.*, vol. 53, pp. 105–115, May 2016.

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 the Department of Electrical and Computer Engineering, New York University (NYU), Brooklyn, NY, USA.

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

Dr. Liu is a recipient of the Ernst Weber Fellowship for his Ph.D. Program in the Department of Electrical and Computer Engineering, NYU.

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 was a Professional Teaching Fellow with the Department of Electrical and Computer Engineering, University of Auckland in 2018. Since 2019, he has been with New York University, Brooklyn, NY, USA, where he is currently a Research Assistant Professor working with the NYU Center for Cybersecurity. His research interests include hardware security, electronic design automation, and machine learning.

Dr. Tan is a member of ACM.

Gaurav Rajavendra Reddy (Member, IEEE) received the Bachelor of Engineering degree from Visvesvaraya Technological University, Belgaum, India, in 2013, and the M.S. and Ph.D. degrees from the University of Texas at Dallas, Richardson, TX, USA, in 2019 and 2020, respectively.

He worked as a Post-Silicon Validation Engineer with Tessolve, Bengaluru, India, from 2013 to 2014. His research interests include applications of machine learning in computer-aided design and design for manufacturability.

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. He is an Associate Professor with New York University, Brooklyn, NY, USA. 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.

Yiorgos Makris (Senior Member, IEEE) received the Diploma of Computer Engineering degree from the University of Patras, Patras, Greece, in 1995, and the M.S. and Ph.D. degrees in computer engineering from the University of California at San Diego, San Diego, CA, USA, in 1998 and 2001, respectively.

After spending a decade on the faculty of Yale University, New Haven, CT, USA, he joined the University of Texas at Dallas, Richardson, TX, USA, where he is currently a Professor of Electrical and Computer Engineering, leading the Trusted and Reliable Architectures Research Laboratory, and the Safety, Security, and Healthcare thrust leader for Texas Analog Center of Excellence. His research focuses on applications of machine learning and statistical analysis in the development of trusted and reliable integrated circuits and systems, with particular emphasis in the analog/RF domain.

Prof. Makris is a recipient of the 2006 Sheffield Distinguished Teaching Award, the Best Paper Awards from the 2013 IEEE/ACM Design Automation and Test in Europe conference and the 2015 IEEE VLSI Test Symposium, as well as Best Hardware Demonstration Awards from the 2016 and the 2018 IEEE Hardware-Oriented Security and Trust Symposia. He serves as an Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS and has served as an Associate Editor for the IEEE INFORMATION FORENSICS AND SECURITY and the IEEE DESIGN & TEST OF COMPUTERS Periodical, and as a Guest Editor for the IEEE TRANSACTIONS ON COMPUTERS and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS.

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 a Professor of ECE with New York University (NYU), Brooklyn, NY, USA, where he co-founded and 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.