Adversarial Perturbation Attacks on ML-based CAD: A Case Study on CNN-based Lithographic Hotspot Detection

KANG LIU, New York University, USA
HAOYU YANG and YUZHE MA, Chinese University of Hong Kong
BENJAMIN TAN, New York University, USA
BEI YU and EVANGELINE F. Y. YOUNG, Chinese University of Hong Kong
RAMESH KARRI and SIDDHARTH GARG, New York University, USA

There is substantial interest in the use of machine learning (ML)-based techniques throughout the electronic computer-aided design (CAD) flow, particularly those based on deep learning. However, while deep learning methods have surpassed state-of-the-art performance in several applications, they have exhibited intrinsic susceptibility to adversarial perturbations—small but deliberate alterations to the input of a neural network, precipitating incorrect predictions. In this article, we seek to investigate whether adversarial perturbations pose risks to ML-based CAD tools, and if so, how these risks can be mitigated. To this end, we use a motivating case study of lithographic hotspot detection, for which convolutional neural networks (CNN) have shown great promise. In this context, we show the *first* adversarial perturbation attacks on state-of-the-art CNN-based hotspot detectors; specifically, we show that small (on average 0.5% modified area), functionality preserving, and design-constraint-satisfying changes to a layout can nonetheless trick a CNN-based hotspot detector into predicting the modified layout as hotspot free (with up to 99.7% success in finding perturbations that flip a detector's output prediction, based on a given set of attack constraints). We propose an adversarial retraining strategy to improve the robustness of CNN-based hotspot detection and show that this strategy significantly improves robustness (by a factor of ~3) against adversarial attacks without compromising classification accuracy.

CCS Concepts: • **Security and privacy;** • **Hardware** → *Physical design (EDA)*; • **Computing methodologies** → *Machine learning*;

Additional Key Words and Phrases: ML-based CAD, security, adversarial perturbations, lithographic hotspot detection

Submitted to the Special Issue on Machine Learning for CAD (ML-CAD).
S. Garg was supported in part by the National Science Foundation (NSF) through the NSF CAREER Award No. 1553419 and NSF SATC Award No. 1801495. B. Tan and R. Karri were supported in part by ONR Award No. N00014-18-1-2058. R. Karri was supported in part by the NYU/NYUAD Center for Cyber Security.
Authors' addresses: K. Liu, B. Tan, R. Karri, and S. Garg, Center for Cybersecurity, New York University Tandon School of Engineering, 370 Jay St, Brooklyn, NY 11201, USA; emails: {kang.liu, benjamin.tan, rkarri, sg175}@nyu.edu; H. Yang, Y. Ma, B. Yu, and E. F. Y. Young, Department of Computer Science and Engineering, SHB913, Chinese University of Hong Kong, Shatin, Hong Kong SAR; emails: {hyyang, yzma, byu, fyyoung}@cse.cuhk.edu.hk.

1084-4309/2020/08-ART48 $15.00
https://doi.org/10.1145/3408288

ACM Reference format:
Kang Liu, Haoyu Yang, Yuzhe Ma, Benjamin Tan, Bei Yu, Evangeline F. Y. Young, Ramesh Karri, and Siddharth Garg. 2020. Adversarial Perturbation Attacks on ML-based CAD: A Case Study on CNN-based Lithographic Hotspot Detection. *ACM Trans. Des. Autom. Electron. Syst.* 25, 5, Article 48 (August 2020), 31 pages.
https://doi.org/10.1145/3408288

1 INTRODUCTION

Electronic system design flows provide several optimization and verification challenges as the scale and complexity of designs increases, placing a higher pressure on designers to deliver timely results. There is substantial interest in using machine learning (ML) techniques for solving hard electronic computer-aided design (CAD) problems ranging from logic synthesis to physical design and design for manufacturability (DFM) [34]. A promised outcome of deep learning enhanced design flows is a faster and scalable development cycle, enabled by improvements in time-consuming steps of design space exploration [16], logic optimization [59], and lithographic analysis [56].

Nonetheless, while deep learning methods have surpassed state-of-of-the-art performance on a wide range of applications, they have been shown to be brittle against adversarial perturbations [14]. Adversarial perturbations are small, imperceptible but targeted modifications to the input of the deep neural network, resulting in incorrect behavior. For example, Figure 1 shows an image of a `horse` from the `CIFAR-10` dataset [23]—each of the the subsequent four images are adversarially perturbed versions of the first that are classified as `airplane`, `automobile`, `bird`, and `cat`, respectively. As noted earlier, the perturbations are so small that they are imperceptible.

Adversarial perturbations have been demonstrated in practically every application in which deep networks are used [3], and have raised fundamental questions about the ability of deep neural networks to generalize. This leads to a natural question: *What are the implications of adversarial perturbations on the security, soundness, and robustness of deep learning techniques in CAD-related problems?* While the CAD domain presents a challenge for adversaries, given the domain-specific knowledge required to perform stealthy (and meaningful) attacks, it is crucial to investigate whether adversarial perturbations pose a potential concern for this innovation.

As a motivating example, we study the challenging CAD problem of lithographic layout hotspot detection. In physical design of an integrated circuit (IC), layout patterns are etched into silicon using optical lithography. Due to lithographic process variations, specific patterns are susceptible to manufacturing errors; these *hotspots* need to be detected and fixed early in the IC design flow to avoid yield loss. The conventional approach to hotspot detection is physics-based optical lithography simulations. While accurate, they are time-consuming and computationally expensive for the full IC. Noting that one can pose hotspot detection as image classification, recent work has proposed adoption of convolutional neural networks (CNN) for this problem, achieving state-of-the-art results [56]. Once hotspots are detected, resolution enhancement techniques (RETs) such as optical proximity correction (OPC) and the insertion of sub-resolution assist features (SRAFs) can enhance IC layouts. Changes are verified using further lithography simulations, and iterated upon as required.

Now consider the following scenario where a designer is considering the purchase of a third-party macro for their IC design. The designer wants to check the quality of the macro and has the IC layout images for verification. Using a CNN-based hotspot detector, the designer can quickly ascertain if the IC layout is printable as-is, and gauge the potential effort needed to correct any design flaws. To pass off a sub-par design as *high quality*, the third-party vendor selectively modifies the layout to force the detector to misclassify hotspot regions as non-hotspot. In other words, the attacker *hides* hotspots by exploiting properties of the CNN—identifying and

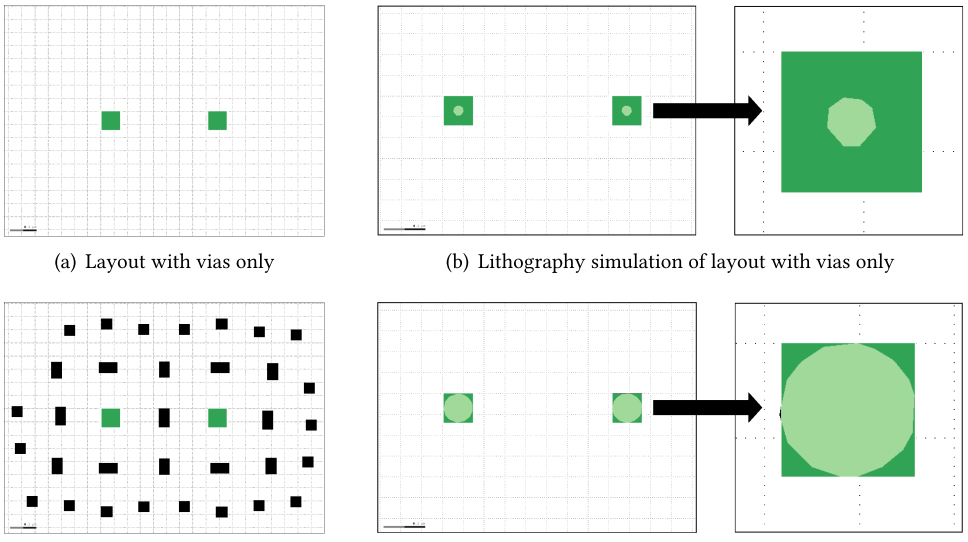
Fig. 1. A "clean" image of a horse (leftmost) and adversarial images with corresponding prediction labels. The adversarial perturbations are so minute as to appear imperceptible.

taking advantage of the susceptibility of CNNs to adversarial perturbations. However, malicious insertion is non-trivial. Unlike image perturbations that involve adding imperceptible noise [3], the attacker must add *semantically meaningful* and realistic IC layout features to the design that pass design rule check (DRC), such as respecting spacing constraints. Successful attacks can have a significant impact: this sabotage can propagate undetected manufacturability issues, causing downstream reductions in IC yield and wasted designer effort.

With lithographic hotspot detection as our motivating case study, we investigate, for the first time, a targeted attack on deep learning-based CAD tools, to demonstrate the feasibility, challenges, and potential security implications for the CAD community. The main goal of this article is to establish whether adversarial input perturbation attacks are possible, given a constraint that perturbations need to be *semantically meaningful*, thus presenting the first insights into how adversarial machine learning might affect ML-CAD. However, while we frame the discussion in this study within a security-related setting (where the CAD flow is compromised by a malicious third-party supplier, as established in prior studies of hardware security threats in the supply chain [40]), our empirical findings motivate a **broader need to study the robustness implications** of integrating ML-based tools into the IC design flow. Our contributions are thus:

- The first exploration of the impact of adversarial perturbations on deep neural network-based CAD tools using IC lithographic hotspot detection as a case study.
- Comprehensive evaluation of two attack scenarios on CNN-based hotspot detectors: (1) white-box attacks, wherein the attacker has access to the model parameters of the detector and (2) black-box attacks, wherein the attacker has access only to the outputs of the detector.
- An initial study into the transferability of adversarial perturbations across CNN-based hotspot detectors with variations of neural network architectures.
- Exploration of adversarial retraining as a defense against adversarial perturbation attacks, yielding an equally accurate but robustified CNN for hotspot detection.

The rest of the article is organized as follows. We explore the motivations for adopting deep learning in hotspot detection, outlining the motivations and goals of a potential attacker (Section 2). This is followed by technical preliminaries to understand the principles of CNNs and the notion of adversarial perturbations (Section 3). In this work, our study centers around two CNN-based hotspot detectors, and we detail the architectures and training of these detectors (Section 4). Following this, we describe the attack methodologies, detailing how adversarial IC layouts can be generated to effectively *hide* the presence of hotspots (Section 5). The first attack is a white-box attack, where the internals of the detector are available to the attacker. We then consider a more conservative attack, where the attacker can only query a black-box model. We verify, via lithography simulations, that a vast majority of adversarially perturbed IC layouts are still hotspots but are not picked up as such by the hotspot detectors (Section 6). Given the high



(a) Layout with vias only (b) Lithography simulation of layout with vias only

(c) Layout with vias and SRAFs (d) Lithography simulation of layout with vias and SRAFs

Fig. 2. Illustration of lithography simulation results of layouts with vias only, and both vias and SRAFs.

success rate of our attack experiments, we propose robust retraining—a promising defense against adversarial attacks—presenting encouraging results (Section 7). Our findings pose interesting questions that we then discuss (Section 8). To contextualize our work, we present related literature in deep learning for CAD and adversarial attacks (Section 9). Ultimately, we conclude from this study that one should be aware of the limitations of using deep learning in CAD, and also be encouraged to investigate and adopt proactive countermeasures (Section 10).

2 MOTIVATION

2.1 Deep Learning for Hotspot Detection

2.1.1 Lithographic Hotspot Detection. In advanced technology nodes the layout feature sizes are much smaller than the light wavelengths used in the optical lithography systems. As a result, complex interactions between light patterns in lithography have made printed patterns sensitive to process variations. This has increased challenges in IC back-end design and sign-off flows. Lithography induces defects due to phenomena such as diffraction, resulting in lithographic *hotspots* [39, 55, 56].

Consider Figure 2(a), which shows an IC layout containing two vias colored green. If this layout were printed as is, then the resulting printed output would be unsatisfactory. Only a small region of the desired vias is printed—shown in light green in Figure 2(b). Thus, resolution enhancement techniques (RETs) such as sub-resolution assist features (SRAF) [13, 53] and optical proximity correction (OPC) [54] have been proposed to ease IC layout manufacturability; they aim to compensate for distortion during lithography. Figure 2(c) shows the effect of SRAF insertion. The printed pattern more accurately reflects the required pattern (Figure 2(d)). However, even when equipped with rigorous RETs, the layout can have hotspots due to unpredictable lithography process variations. Therefore, it is vital to spot potential hotspots before manufacturing and correct them either by using RET or by re-design.

2.1.2 Deep Learning-based Hotspot Detection. In light of the prohibitive run-time of lithographic simulation, recent work has sought to speed-up hotspot detection using pattern matching

[60] and machine learning [30, 57]. Pattern matching methods find similar or identical hotspot-causing patterns in a new design from a library of known hotspots. These techniques are both fast and accurate if the patterns are similar to those in the library, but cannot find previously unseen hotspot patterns. In contrast, machine learning solutions seek to capture the underlying *physics* of lithographic simulation (i.e., the relationships between IC layout features and their manufacturability) and, as such, *generalize* to unseen patterns (or at least that has been the hope). Recent advancements on CNN-based hotspot detection [55, 56] have shown that both shallow and deep CNNs are more accurate compared to legacy machine learning-based and pattern matching-based techniques.

2.2 Threat Model

2.2.1 Setting. To motivate our work in examining the security and robustness of deep learning in CAD, we explore the scenario of a designer considering the purchase of a macro from a third-party IP vendor, as posed in previous studies of threats to the hardware supply chain [40]. In the threat model, the third-party IP vendor distributes hard macros in GDSII format [9], where the circuit is laid out and allegedly enhanced for lithography using RETs. As part of the validation process, the designer does a "sanity-check" on the macro to establish its quality by using a CNN-based hotspot detector (which may be a commercial tool in a local or cloud setting).

2.2.2 Attack Goals. The vendor aims to sell low-quality hard macros, either to make a profit from their design short-cuts or to sabotage the designer (by forcing them to waste time and resources in rectifying poor designs). To achieve this aim, the attacker's goal is, therefore, *to fool the target CNN-based hardware detector into classifying hotspots as non-hotspots.* This should be achieved by making the smallest changes to the layouts as possible. In this work, we investigate SRAF insertion as an RET; consequently, the aim of the attacker is to insert as few SRAFs as possible, to make the whole design appear to be free of hotspots.

2.2.3 Attacker Capabilities. In the context of deep learning, the attacker capabilities can be defined based on the amount of information they possess about the network under attack. This includes information about the network's hyper-parameters: its overall architecture, its weights and biases, the training algorithms and training data, and so on. For this case study, we consider two scenarios: (1) an attacker with white-box access, where they have full knowledge of the CNN, including its network architecture, weights and biases; and (2) an attacker with only black-box access, where they are able to query the detector, receiving both output classification as well as the accompanying prediction confidence. Both models have been studied in prior work [3, 14, 27, 38].

3 DEEP LEARNING PRELIMINARIES

To appreciate the potential of deep learning for CAD problems such as hotspot detection, we present relevant technical preliminaries for CNNs and adversarial perturbations.

3.1 CNN Basics

A CNN features an input layer, a number of hidden layers, and an output layer. The CNN takes in some input (e.g., an image) and propagates the data through a series of linear and non-linear operations (akin to convolution and activation of "neurons"). After all the input has been transformed by each of the hidden layers, the final output produces a classification prediction for the input. The CNN is "trained" by configuring the parameters of the filters in each layer (the weights).

We can express this formally as follows. A neural network is defined as a function F that takes input $x \in \mathbb{R}^N$ and gives output $z \in \mathbb{R}^m$, such that $z = F(x)$. For an m-class classifier, we define z as an array, $z = [z_1, z_2, \ldots, z_m]$, where z_i is the prediction probability of class i, $i = 1, 2, \ldots, m$. The

network output z is subject to the constraints: $0 \leq z_i \leq 1$, and $z_1 + z_2 + \cdots + z_m = 1$. The label y of input x takes the output class with the highest prediction probability, such that $y = \text{label}(x) = \arg\max_i z = \arg\max_i F(x)$. A deep neural network classifier has multiple layers of neurons, the last being a softmax layer. Hence, the neural network can be expressed as

$$F = \text{input layer} \circ F_1 \circ F_2 \circ \cdots \circ F_{k-1} \circ F_k \circ \text{softmax}, \tag{1}$$

where

$$F_i(x) = f_i(w_i x + b_i), \quad i = 1, 2, \ldots, k. \tag{2}$$

Here f_i is the activation function of layer F_i, and w_i is the model weights and b_i is the bias. Some common choices of activation function f include logistic, tanh, and ReLU [36]. In an image classification neural network, input x is either a grey-scale image with one channel or an RGB image with three channels, where each channel of pixel x_i takes integer values from $[0, 255]$.

3.2 Adversarial Perturbations

The existence of adversarial inputs for classification using neural networks was first described by Szegedy et al. [43]. They observed a phenomenon whereby neural networks would change its output prediction based on imperceptible perturbations in the input. In these cases, while the network would be "fooled," a human would not be "fooled." This property can be exploited by an adversary, whereby inputs can be crafted to fool a target network and cause misclassification.

Formally, let y^* denote the true label of a clean input x, and y denote the prediction label of x given by the neural network. The adversary aims to generate an adversarial input x' close to x, and mislead the network to output a target label y', while $y' \neq y$. The difference between x and x' is measured by a distance metric and constrained by a constant δ, such that $\|x - x'\| \leq \delta$. Normally δ is so small to be perceptual to human eyes and should not change the prediction label from y to y'.

In non-targeted attacks, adversaries search for adversarial inputs x' as long as its output label $y' \neq y$. In targeted attacks, the target label is pre-defined by the adversary, and y' could be quite distinct than y. There are several schemes for crafting adversarial perturbations. Our work is inspired by the following methods that have been explored in a general adversarial perturbation context.

3.2.1 Basic fast Gradient Sign (FGS) Method. Goodfellow et al. [14] proposed the FGS method for adversarial input generation. For non-targeted attacks, starting with a clean input x, the adversary moves each pixel in the opposite direction of the gradient of the loss function of the true label with respect to x. The goal is to mislead the network into outputting any label other than the true label. The non-targeted FGS attack can be described mathematically as follows:

$$x' \leftarrow \text{clip}(x + \epsilon \text{sign}(\nabla \ell_{F,y^*}(x))). \tag{3}$$

ϵ is a small constraint scalar, ℓ is the loss function and $\text{clip}(x)$ ensures pixel values fall in the desired range.

However, in a targeted attack, the adversary seeks to fool the network into misclassifying x as a specific target label. This is achieved by altering pixels in the direction of the gradient of the loss function of the target label with respect to x. The attack is described by Equation (4):

$$x' \leftarrow \text{clip}(x - \epsilon \text{sign}(\nabla \ell_{F,y'}(x))). \tag{4}$$

These two attacks emphasize computational efficiency and speed at the expense of introducing relatively large perturbations. Sophisticated techniques that seek to find the smallest possible perturbation, albeit at greater computational expense, have subsequently been proposed [24]—one such attack is described next.

3.2.2 Iterative Fast Gradient Sign (IFGS) Methods. IFGS methods operate over multiple iterations, adding relatively small perturbations in each [24]. As such, IFGS methods can generate adversarial inputs with smaller distortion when compared to basic FGS. Equations (5) and (6) describe the updates performed by the non-targeted and targeted versions of IFGS in each iteration. As an example, the adversarial perturbations in Figure 1 were generated by the IFGS method:

$$x'_0 = x, \quad x'_{N+1} \leftarrow \text{clip}_\epsilon(x'_N + \alpha \text{sign}(\nabla \ell_{F,y^*}(x))), \tag{5}$$

$$x'_0 = x, \quad x'_{N+1} \leftarrow \text{clip}_\epsilon(x'_N - \alpha \text{sign}(\nabla \ell_{F,y'}(x))). \tag{6}$$

3.2.3 Semantically Meaningful Perturbations. Another body of work has focused on semantically meaningful perturbations. For instance, specially crafted stickers affixed to traffic signs can mislead traffic sign classifiers [12]. These perturbations are not imperceptible, in fact, quite the opposite, they are easily spotted, but are designed to seem innocuous. For instance, a human is unlikely to think that a small sticker on a traffic sign indicates an adversarial attack. Our work crafts such perceptible but semantically meaningful perturbations. However, the notion of what is semantically meaningful is informed by the underlying domain of lithography and physical layout design.

4 CASE STUDY: IC LITHOGRAPHIC HOTSPOT DETECTION

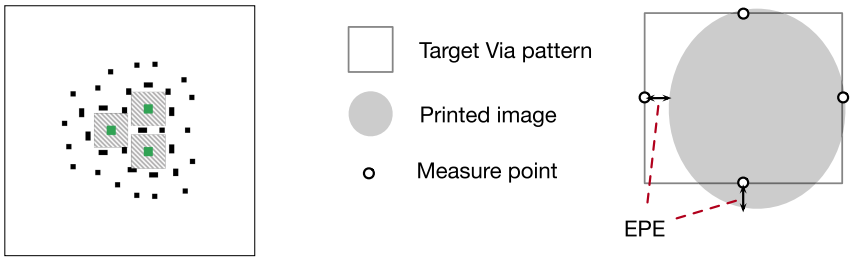
In this work, we use two different CNN-based hotspot detectors to explore our proposed attacks. They are trained using the same dataset and act as *targets* for adversarial perturbations. This section describes details of our dataset, the network architectures, and the training process. Our case study draws heavily from prior state-of-the-art work [56].

4.1 Layout Dataset

Existing datasets for lithographic hotspot detection, for example, the widely used ICCAD'12 contest dataset [46], do not come with much of the information required to verify the success of adversarial attacks. For instance, the ICCAD'12 data specifies neither design rules nor does it specify lithography simulation parameters. Therefore, for this case study, we prepared our own layout dataset comprising 10,403 layout clips stored in the GDSII format. We targeted the detection of lithographic hotspots for via layers using SRAF-based RET. To create the large number of layout samples, we generated the via patterns in the following manner:

(1) Within each clip region ($2 \times 2\mu$m), we place lower layer metal gratings with fixed wire critical dimension (CD) and pitch;
(2) We add an upper metal layer with preset CD and spacing constraints;
(3) The cross regions between two metal layers become candidates for via placement—we place vias stochastically with a given probability;
(4) Finally, vias that violate design rules are removed. In this dataset, we use vias sized 70×70 nm and enforce a minimum via spacing of 70 nm.

Once the "raw" layouts are produced, we perform optical proximity correction (OPC) and lithography simulation using Mentor Calibre [15] to insert SRAFs; we set the allowable SRAF region to a 100–500 nm city-block distance. It should be noted that we adopt configurations of directprint DUV technology node for our generated via patterns with soft-annular illumination and the process technology is also verified in recent works [13, 53]. An example of a layout clip is shown in Figure 3(a). Next, we determine the ground truth hotspot/non-hotspot labels for the layouts. In this work, we use the edge placement error (EPE) as our metric for determining the quality of the printed patterns. Each via pattern in a layout is associated with four measure points, with



(a) Example Layout (b) Edge Placement Error (EPE)

Fig. 3. (a) Example of a layout with vias (in green), SRAFs (in black), and the forbidden areas (striped region). (b) Illustration of edge placement error (EPE). Each target via pattern has four measure points (one at the center of each edge). The EPE is the perpendicular displacement from the measure point to the corresponding printed image (contour).

one point at the center of each edge. The EPE is defined as the perpendicular displacement from the measure point to the corresponding printed contour, as illustrated in Figure 3(b). A layout is identified as a hotspot layout if there exists any measure points with the EPE greater than 2 nm, as in typical industrial settings.

As shown in Figure 3(a), the layouts we produced have three key features: vias (the desired pattern to be printed), SRAFs (used to improve printability), and forbidden regions (where SRAFs should not be placed). Each via is surrounded by a square forbidden region whose edges are 100 nm from the via's edges. The GDSII files contain three layers of interest: (1) a via layer, (2) an SRAF layer, and (3) a forbidden region layer.

4.2 Design of CNN-based Hotspot Detectors

Using the layout dataset, we trained two different CNN-based hotspot detectors to represent networks of different complexity, adopting procedures described in prior work [56]. The parameters are shown in Tables 1 and 2.

- *Network A* is a smaller 9-layer network that is fast(er) to train and for prediction. We observed that further increasing network depth/complexity did not increase accuracy; i.e., Network A is "right-sized" for accuracy.
- *Network B* is a larger 15-layer network that is slower to train, but is potentially less susceptible to attack as the complex architecture learns sophisticated features for hotspot detection. Prior work on adversarial robustness suggests that deeper, complex network are more resilient to attack [29].

4.2.1 Data Preprocessing. The dimension of the GDSII layouts is 2,000 × 2,000 nm, which can be represented as 2,000 × 2,000 pixel binary-valued images, where all the layers are flattened. Layout polygons are represented with pixel intensity of 255 and the background is represented with a pixel intensity of 0. For training and inference, we scale the layout image by a factor of 255 so that all the pixel intensities are either 1 or 0.

Training a CNN on large images requires significant computation resources and time. Therefore, as proposed in Reference [56], we compute a discrete cosine transformation (DCT) on each image to extract its features as input for the networks. The equation for DCT is shown in Equation (7):

$$D_{k_1,k_2} = \sum_{n_1=0}^{N_1-1} \sum_{n_2=0}^{N_2-1} I_{n_1,n_2} \cos\left[\frac{\pi}{N_1}\left(n_1+\frac{1}{2}\right)k_1\right] \cos\left[\frac{\pi}{N_2}\left(n_2+\frac{1}{2}\right)k_2\right]. \tag{7}$$

Table 1. Architecture of Network A

Layer	Kernel Size	Stride	Output Size
input	-	-	(20, 20, 32)
conv1_1	3	1	(20, 20, 16)
conv1_2	3	1	(20, 20, 16)
maxpooling1	2	2	(10 , 10, 16)
conv2_1	3	1	(10, 10, 32)
conv2_2	3	1	(10, 10, 32)
maxpooling2	2	2	(5, 5, 32)
fc1	-	-	250
fc2	-	-	2

Table 2. Architecture of Network B

Layer	Kernel Size	Stride	Output Size
input	-	-	(20, 20, 36)
conv1_1	3	1	(20, 20, 16)
conv1_2	3	1	(20, 20, 16)
conv1_3	3	1	(20, 20, 16)
maxpooling1	2	2	(10, 10, 16)
conv2_1	3	1	(10, 10, 32)
conv2_2	3	1	(10, 10, 32)
conv2_3	3	1	(10, 10, 32)
maxpooling2	2	2	(5, 5, 32)
conv3_1	3	1	(5, 5, 64)
conv3_2	3	1	(5, 5, 64)
conv3_3	3	1	(5, 5, 64)
maxpooling3	2	2	(3, 3, 64)
fc1	-	-	500
fc2	-	-	2

Table 3. Confusion Matrix of Networks A and B

		Prediction			
		Network A		Network B	
		non-hotspot	hotspot	non-hotspot	hotspot
Condition	non-hotspot	0.72	0.28	0.72	0.28
	hotspot	0.29	0.71	0.28	0.72

Here n_1 and n_2 are the horizontal and vertical coordinates of the image pixels, and N_1 and N_2 are the width and height of the image. k_1 and k_2 represent the horizontal and vertical coordinates of the DCT coefficients. To reduce the image dimensions, we perform DCT on non-overlapping 100 pixel $\times$ 100 pixel sub-blocks on each layout image (with a 100 pixel stride), and then keep a selection of DCT coefficients. For Network A, we keep the coefficients of the 32 lowest frequencies, producing inputs of size (20, 20, 32). For Network B, we keep the coefficients of the 36 lowest frequencies (i.e., more information for the larger network), producing inputs of size (20, 20, 36). This speeds up training with low loss of information without affecting network performance.

4.2.2 Training. We train both networks using the same layout dataset. We randomly split 10,403 layout images into 8,000 training images and 2,403 test images, where the training data consists of 2,774 hotspot and 5,226 non-hotspot images, and test data has 841 hotspot and 1,562 non-hotspot images. To compensate for data imbalance, we incorporate class weights to weigh the loss function during training, which tells the model to "pay more attention" to samples from an under-represented class [18]. This is done for both networks to achieve a balanced hotspot and non-hotspot detection accuracy. We implement network training with the Keras library [8], and use the ADAM optimizer [21] for loss minimization. The confusion matrix is shown in Table 3. Our training of the baseline networks follows the same methodology as in prior work [56]. Although Network A and Network B have the same overall[1] and hotspot prediction accuracy, we seek to explore the robustness of both networks with different depths/complexity.

[1]Overall accuracy is defined as the average of non-hotspot classification accuracy and hotspot classification accuracy.

The techniques used to train the hotspot detectors are comparable to prior works in lithographic hotspot detection; we used the state-of-the-art architecture, data processing, and training procedures as detailed in Reference [56]. The baseline accuracy that one can achieve depends on a mixture of factors, including the nature of the dataset and how the designer may choose to bias their network (for example, hotspot detection accuracy could be prioritized, possibly at the expense of high false positives or poorer non-hotspot accuracy). Thus, in this work, we aimed for balanced overall (i.e., both hotspot and non-hotspot classification) accuracy. While our detectors have lower hotspot detection accuracy compared to the results presented in Reference [56], we use a completely different dataset (enabling us to perform lithography simulation after performing the adversarial attack). When compared to other prior work, our detectors have better overall accuracy compared to the "SOTA" (without data augmentation) as reported in Reference [39] (we achieve ~72% accuracy, while their reporting of the SOTA performance on their dataset was ~70.6%), and achieve better "non-hotspot hit rate" (we achieve ~72% non-hotspot accuracy vs. their 41.50%). The difference between training and test data accuracy is <3%, suggesting no overfitting. In this work, our main focus is on investigating whether the phenomenon of adversarial perturbations exists in a CAD-domain problem—experimental evaluation of other DL architectures remains future work.

5 PROPOSED ATTACK METHODOLOGIES

5.1 Overview

We propose attack methodologies for modifying layouts with hotspots such that they fool the CNN-based hotspot detector into misclassifing layouts as non-hotspot. We experiment with two attack types: a white-box attack, where the attacker has full access to the internal details (weights, architecture, etc.) of the hotspot detector, and a black-box attack, where the attacker can only query the detector to receive the output prediction and associated confidence. During the attack, the attacker aims to fool the target detector by modifying layouts in a *semantically meaningful* way. This means that the attacker cannot alter the IC layout by moving via locations as this may change design functionality; in this attack, we only add SRAFs to the layout. Further, the modifications must be small and innocuous, for instance, by only using shapes that already exist in the layout dataset. Finally, the perturbations should not introduce DRC violations. Based on these considerations, our perturbations must satisfy the following constraints:

(1) Insertion Constraint: Maliciously-inserted SRAFs can only be added to the SRAF layer.
(2) Shape Constraint: Maliciously-inserted SRAFs should be rectangles, with a fixed width of 40 nm. The height can be selected within 40–90 nm, at a resolution of 1 nm. The SRAF can be placed either horizontally or vertically.
(3) Spacing Constraint: The Euclidean distance between any two SRAFs should be at least 40 nm.
(4) Forbidden Zone Constraint: Maliciously-inserted SRAFs cannot overlap with the forbidden region in a layout.

For simplicity, the our attack evaluation involves adding 40 nm wide SRAFs with the following height options: 40, 50, 60, 70, 80, or 90 nm, all placed horizontally. These design rules are compatible with legacy 32~45 nm nodes. A similar public layout example can be found in the Nangate 45 nm Open Cell Library [22].

5.2 White-box Attack

In the white-box attack, the attacker knows the internal details of the target CNN-based hotspot detector, and exploits this as part of the attack. We propose a gradient-guided approach to generate

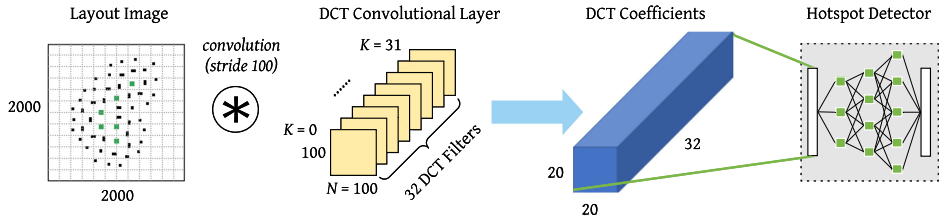


Fig. 4. Illustration of end-to-end hotspot detection with DCT implemented as a convolutional layer.

adversarial layouts, inspired by the fast gradient sign approach [14] (explained in Section 3). Since the baseline hotspot detection networks (in Section 4.2) take DCT coefficients as inputs, a naïve attack would need to modify these coefficients and then perform inverse-DCT to produce adversarially perturbed layouts. There are at least three reasons why this naïve approach is infeasible:

(1) There is not enough data to reconstruct any layout without information loss, since the input DCT coefficients used for inference are only the low frequency components.
(2) There is no guarantee that modifications of DCT coefficients, when reflected back to layout images, satisfy the attack constraints above.
(3) It is challenging to modify DCT coefficients that result in an exact $0 \rightarrow 1$ change in layout image pixels, as the images are binary-valued.

5.2.1 DCT as a Convolution Layer in the Network. Our solution to this problem is to implement the DCT computation as a convolution layer of the neural network such that the combined network works in an end-to-end fashion. The network takes in layout images as inputs; this allows us to perturb image pixels while also incorporating the attack constraints. This idea comes from Equation (7), where we observed that the summation and element-wise product can be realized directly as a convolution layer of the CNN (without adding bias). The weights of a DCT filter for calculating the Kth DCT coefficient is obtained as shown in Equation (8):

$$W_K = \cos\left[\frac{\pi}{N_1}\left(n_1 + \frac{1}{2}\right)k_1\right]\cos\left[\frac{\pi}{N_2}\left(n_2 + \frac{1}{2}\right)k_2\right]. \tag{8}$$

Here k_1 and k_2 are the horizontal and vertical coordinates of the Kth DCT coefficient, and n_1 and n_2 are the horizontal and vertical coordinates of each weight of the filter. N_1 and N_2 are the width and height of the filter. Since the DCT computation operates on 100×100 sub-blocks of each image, the DCT convolution layer will have filters of size (100, 100) and strides of 100. We illustrate this end-to-end network that combines the DCT computation and hotspot detection in Figure 4.

5.2.2 Attack Process. With this new end-to-end network, the attacker can now explore the gradients of the network in terms of the image and guide placement of SRAFs to positions that have the highest impact on the network output prediction (shifting from hotspot to non-hotspot). We define the loss of the attack as the *distance* between the prediction probability of perturbed hotspot layout and *ideal* non-hotspot layout (i.e., a layout with perfect prediction probability of [1, 0]). This is represented as Equation (9) where $P_{hotspot}(x)$ is the probability that a layout x is classified as hotspot:

$$loss\_adv(x) = (1 - P_{hotspot}(x) - 1)^2 + P_{hotspot}(x)^2 = 2P_{hotspot}(x)^2. \tag{9}$$

The attacker aims to keep minimizing the loss as they choose and add perturbations (i.e., SRAFs) iteratively, until at some point the perturbed layout image is predicted as non-hotspot. As an

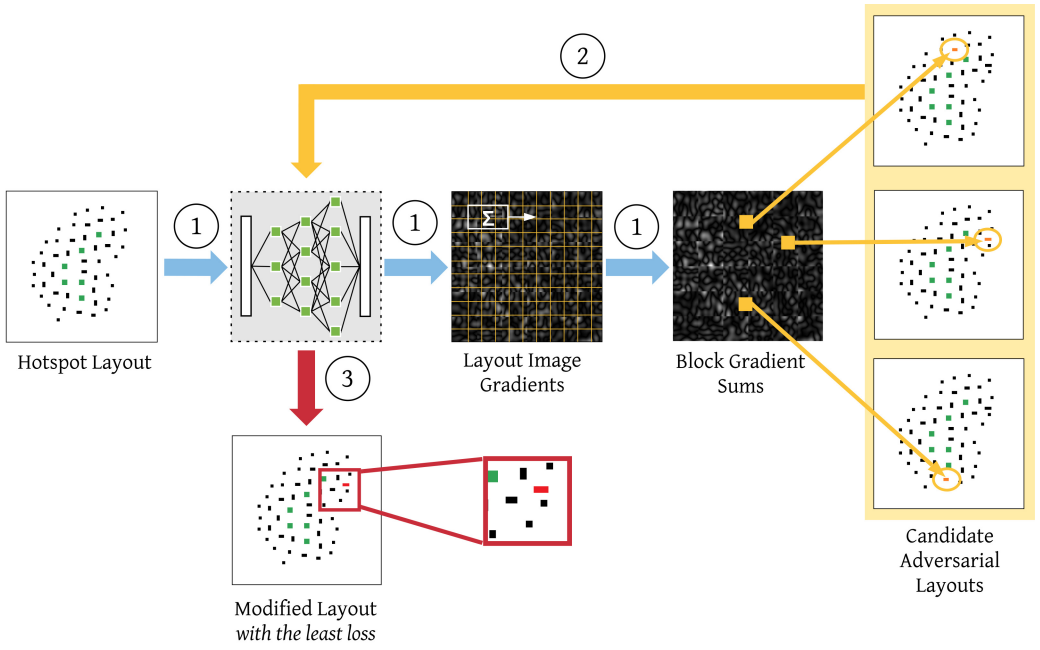


Fig. 5. Illustration of the white-box attack process with one SRAF insertion.

attack parameter the attacker can choose a maximum number of SRAFs to insert, T. The detailed algorithm of the white-box attack is shown in Algorithm 1.

To perform the attack, we first calculate the gradient of the loss function with respect to each pixel (line 4 in Algorithm 1). This gradient represents the amount of "influence" that a given pixel has on the final network prediction. However, since we are modifying *blocks* of image pixels instead of a singe pixel (when we insert SRAFs), we sum the gradients of a potential perturbation block at each potential insertion location (lines 5–10 in Algorithm 1). We illustrate this concept as ① in Figure 5. This represents the "influence" that a perturbation has on the final network prediction when it is inserted in that location. Specifically, we are changing blocks of pixel values in the positive direction from 0 to 1, so the image block that has the largest negative sum of gradients will have the most influence in minimizing the loss.

However, these gradient sums only reflect the influence for a *small* change in the input. As we are shifting pixel values with a relatively large step (i.e., from 0 to 1) there is no guarantee that the largest negative sum of gradients will still have the most significant influence. Therefore, instead of picking the SRAF insertion that has the largest negative sum of gradients, we query the CNN for top-n candidate perturbation blocks with the largest negative sum of gradients (lines 11–14 in Algorithm 1). We refer to this n as the attacker-specified *check parameter*. Of these candidates, we pick the one that has the largest influence (i.e., that fools the network toward predicting a hotspot as non-hotspot) (line 15 in Algorithm 1).

Our strategy for finding candidate SRAF insertion (② in Figure 5) is as follows. We define the location of SRAFs by the coordinate of its top-left corner point. We slide the SRAF over the center region of the layout (we leave a 200 nm boundary on each side of the layout image) with horizontal and vertical stride of 40 nm. This forms all the possible locations for potential perturbation addition. However, if any part of a location and its surrounding 40 nm has an existing pixel value of 1 (i.e., it is already occupied with an SRAF), or overlaps with any forbidden region,

ALGORITHM 1: White-box Attack

1: Input: original hotspot image x, white-box network function F with DCT convolutional layer at the bottom, loss function L, image pixel indexing function P, perturbation pattern set $S = height \times [width1, width2, \dots]$, surrounding spacing d, maximum number of perturbation addition T, check parameter n.
2: $t = 0$
3: **while** $t < T$ **do**
4: compute image gradient $x\_grad = \mathrm{d}L/\mathrm{d}x$
5: **for** *shape* in S **do**
6: **for** each position *pos* in x **do**
7: **if** pixel values of *shape* at *pos* and its surrounding area $P(x, shape, pos, d) = 0$ **then**
8: $sum\_grad[shape, pos] = \sum P(x\_grad, shape, pos)$ ▷ Sum gradients of the *shape* area.
9: **else**
10: $sum\_grad[shape, pos] = \infty$
11: **for** $i = 1$ to n **do**
12: Get *shape* and *pos* of the ith smallest element of sum_grad
13: perturbed image $x' = x$ and set $P(x', shape, pos) = 1$
14: compute loss $loss\_adv[i] = L(x')$
15: $shape,\ pos = \arg\min loss\_adv$ and set $P(x, shape, pos) = 1$ ▷ Insert perturbation.
16: **if** $F(x)$ is hotspot **then**
17: $t = t + 1$
18: **else if** $F(x)$ is non-hotspot **then**
19: **Return**: x ▷ Adversarial non-hotspot layout generated.
20: **Return**: *null* ▷ Otherwise, failed to generate non-hotspot layout within attacker-specified bound.

this location is marked as invalid for SRAFs. We set the loss of this location/shape pair to be ∞ (line 10 in Algorithm 1). In this way, we ensure that inserted SRAFs satisfy the attack constraints.

If the constraints are all satisfied, then we consider this location to be valid. If the sum of the gradients at this location is one of the n largest negative sums, then we compute the loss for this layout image with the hypothetically inserted SRAF shape at this location (shown as ③ of Figure 5). Since the attacker has the flexibility to add six different shapes of SRAFs (width varies from 40, 50, 60, 70, 80, to 90 nm), they will iterate the gradient summation on all the possible locations for each of these perturbation shapes. In each iteration of adding one SRAF, one of the six shapes is added to the current layout such that it yields the lowest prediction probability for hotspot.

The algorithm stops either when the network predicts the perturbed layout as non-hotspot (hotspot prediction probability ≤ 0.5), or when the number of inserted SRAFs has reached the maximum allowance and no adversarial non-hotspot layout is generated (lines 16–20 in Algorithm 1).

5.3 Black-Box Attack

This attack explores the case where an attacker has less knowledge of the target network. With a black-box access to the network, the attacker can query the hotspot detector with computed DCT coefficients of a layout to obtain the output prediction probability. We illustrate the attack in Figure 6. Details of the black-box algorithm are shown in Algorithm 2.

At a high-level, the black-box attack iteratively queries the detector with different SRAF shape and insertion location combinations. The attacker first adds a single SRAF. The attacker exhaustively examines all the possible valid locations for each valid SRAF shape (① in Figure 6), and queries the network with DCT coefficients of the candidate modified layout (② in Figure 6,

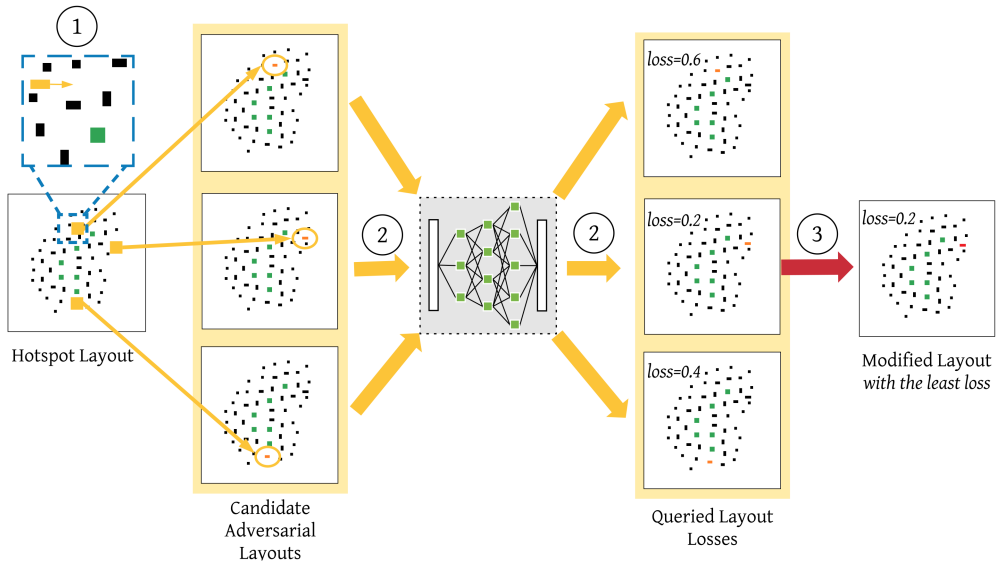


Fig. 6. Illustration of the black-box attack process with one SRAF insertion.

lines 4–10 in Algorithm 2). The location and perturbation that has the minimum loss is selected, using the same loss function as in Equation (9) (③ in Figure 6, line 11 in Algorithm 2). Further SRAFs are added in the same way. Like the white-box attack, the algorithm terminates either by returning a successful adversarial non-hotspot layout, or fails to produce an adversarial layout within the specified maximum number of inserted SRAFs.

6 EMPIRICAL EVALUATION OF ATTACK SUCCESS

6.1 Experimental Setup

To investigate the implications of our proposals, as well as study the relationship between the attack efficacy and the target networks, we performed white-box and black-box attacks on both CNN-based hotspot detection networks. We run all experiments on a desktop computer with Intel CPU i9-7920X (12 cores, 2.90 GHz) and single Nvidia GeForce GTX 1080 Ti GPU. As a measure of the CNN-based hotspot detectors' sensitivity to adversarial input perturbation, we explore how feasible it is to change the output classification by characterizing our proposed attacks along several dimensions: the percentage of hotspot clips where the output classification is flipped after successful perturbation, the average number of added SRAFs needed to trigger the label flip, the average area changed by the SRAF addition (as a percentage of the layout clip), and the average time taken by the proposed attack algorithms for creating adversarial examples. For our analysis, we report the *perturbation success rate*, defined as follows:

Definition 6.1 (Perturbation Success Rate). The perturbation success rate is the percentage of originally hotspot layouts that were successfully perturbed (under attacker-specified constraints, such as maximum number of SRAF insertions) such that they are classified as non-hotspot by the CNN-based hotspot detector.

In Section 6.4, we further investigate if the successfully perturbed layout clips remain truly hotspot (i.e., that they were not incidentally "fixed" by the attack process), thus investigating the "true attack success rate."

Table 4. Summary of Attack Results for White-box and Black-box Algorithms

Attack	White-box		Black-box	
Network	A	B	A	B
Perturbation success rate	99.7%	85.5%	99.7%	93.3%
Average attack time per layout	8.6 s	45.1 s	350.5 s	677.3 s
Average number of SRAFs added	5.3	8.3	4.1	7.3
Average area of SRAFs added	0.3%	0.5%	0.3%	0.5%

For both attacks, the maximum number of SRAF insertions allowed (T) is 20. For the white-box attack, the check parameter (n) is 180. T and n are attacker-specified parameters, as explained in Algorithms 1 and 2.

ALGORITHM 2: Black-box Attack

Input: original hotspot image x, DCT computation function DCT, black-box network function F, loss function L, image pixel indexing function P, perturbation pattern set $S = height \times [width1, width2, \dots]$, surrounding spacing d, maximum number of perturbation additions T.
$t = 0$
while $t < T$ **do**
  for *shape* in S **do**
    for each position *pos* in x **do**
      if pixel values of *shape* at *pos* and its surrounding area $P(x, shape, pos, d) = 0$ **then**
        perturbed image $x' = x$ and set $P(x', shape, pos) = 1$
        compute loss $loss\_adv[shape, pos] = L(DCT(x'))$
      else
        $loss\_adv[shape, pos] = \infty$
  $shape, pos = \arg\min loss\_adv$ and set $P(x, shape, pos) = 1$ ▷ Insert perturbation.
  if $F(DCT(x))$ is hotspot **then**
    $t = t + 1$
  else if $F(DCT(x))$ is non-hotspot **then**
    Return: x ▷ Adversarial non-hotspot layout generated.
Return: *null* ▷ Otherwise, failed to generate non-hotspot layout within attacker-specified bound.

Across all the experiments in this section, we consider a layout to be hotspot if the network prediction probability for hotspot is ≥ 0.5. For the white-box attack, we adversarially perturb 500 correctly classified hotspot layouts from the validation set (i.e., the layouts that were not used for training). As the black-box attack takes longer to perform, we generate adversarial non-hotspot layouts for 150 hotspot layouts. The average attack time is the end-to-end time (including querying the hotspot detector). We limit the maximum number of adversarial SRAF additions to 20, and the check parameter in the white-box attack is 180[2]. We illustrate a selection of attack outputs and their corresponding verification results in Figures 8 and 9. We present a summary of the results in Table 4.

6.2 Attack Results

The most successful attack was on Network A, where we achieved a 99.7% perturbation attack success (498 hotspot layouts were made to be classified as non-hotspot by the CNN). The white-box perturbation success rate drops between our attack of Network A and B (a decrease of 14.2%).

[2]This corresponds to ~11% of the total number of possible SRAF insertion candidates on average.

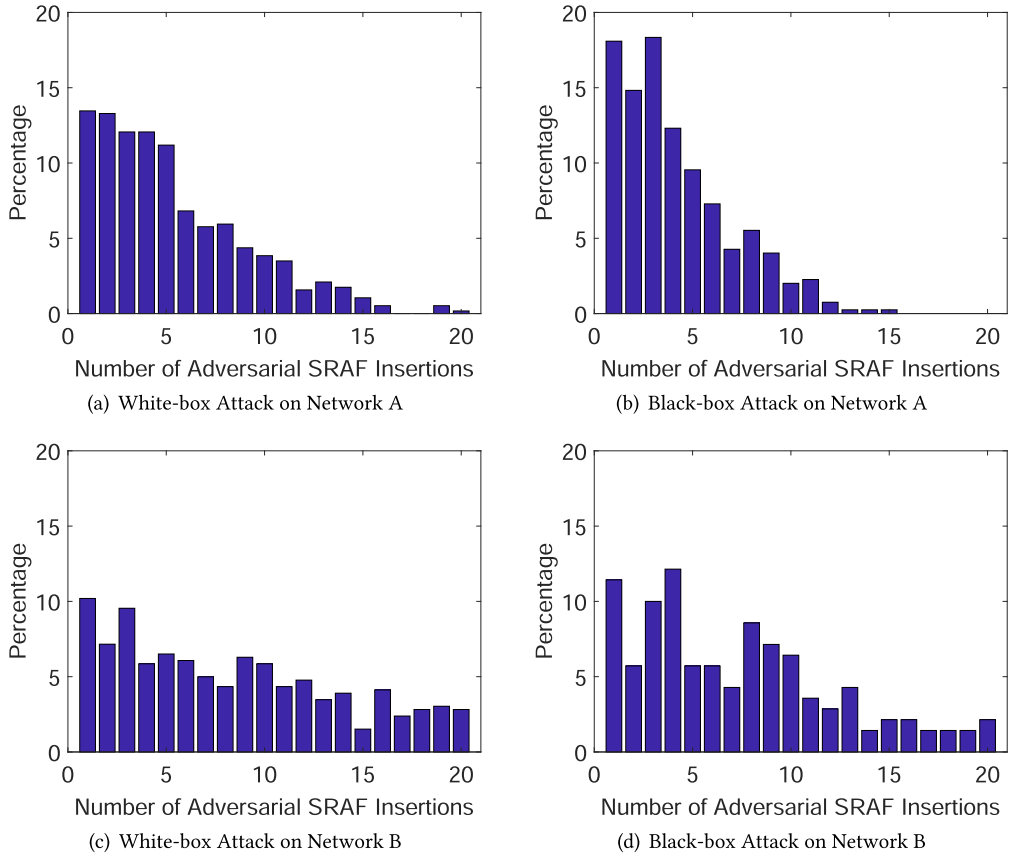
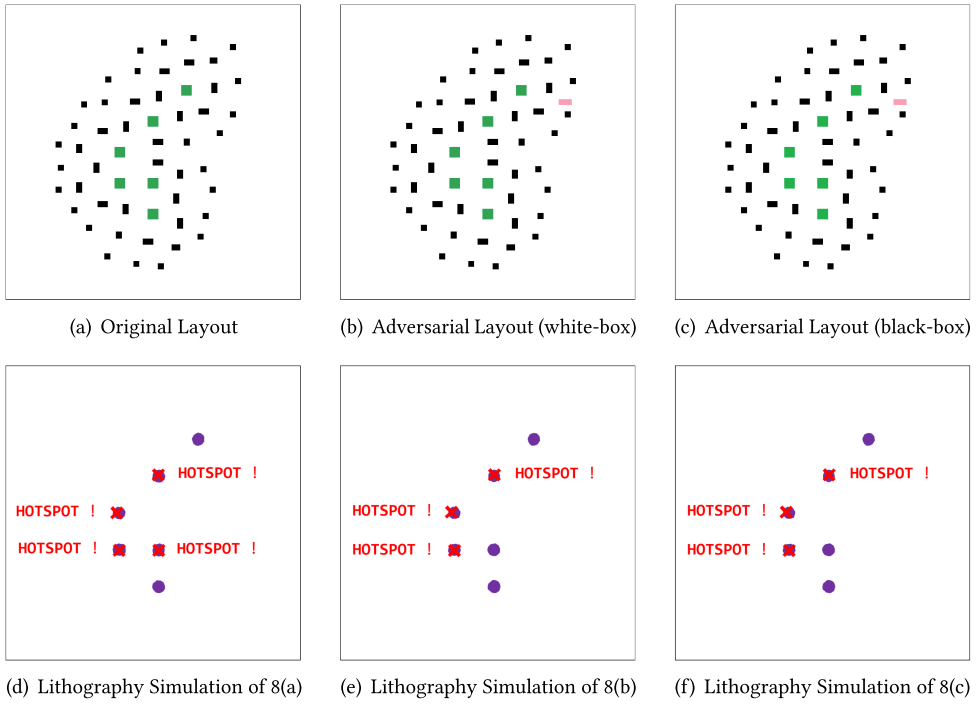


Fig. 7. Histograms of percentages of different number of adversarial SRAF insertions by white-box attack on Networks A (a) and B (c). Histograms of percentages of different number of adversarial SRAF insertions by black-box attack on Networks A (b) and B (d).

One explanation is that the complex Network B has learned more about the characteristics of hotspots, and therefore is more challenging to fool; this is consistent with prior findings of neural network robustness [29].

These trends can be observed in the average time taken to generate an adversarial layout in the white-box attack. The average attack time is 8.6 s/45.1 s for Network A and B, respectively, in the white-box attack. Network B required on average ~ 6× more time than the white-box attack on Network A. The extra time for the white-box attack on Network B to produce a successful perturbed layout is partially due to the increased feedforward computation on more layers (higher overhead in query-time during the attack). Similarly, the average number of SRAFs inserted is greater for the more complex Network B compared to the simpler Network A.

Figures 7(a) and 7(c) show the percentage of layouts that were successfully perturbed by a given number of SRAF insertions for the white-box attack. In all cases, the minimum number of SRAFs that needed to be added to cause misclassification was 1, and an example of this is shown in Figure 8. Of the layouts that were successfully perturbed to appear as non-hotspot in each attack, ~13% required only one inserted SRAF to fool Network A, and ~10% for Network B. Furthermore, 50% of the perturbed layouts could fool Network A with 4 or fewer inserted SRAFs. For Network B, 50% of the perturbed layouts had seven or fewer inserted SRAFs.



(a) Original Layout (b) Adversarial Layout (white-box) (c) Adversarial Layout (black-box)

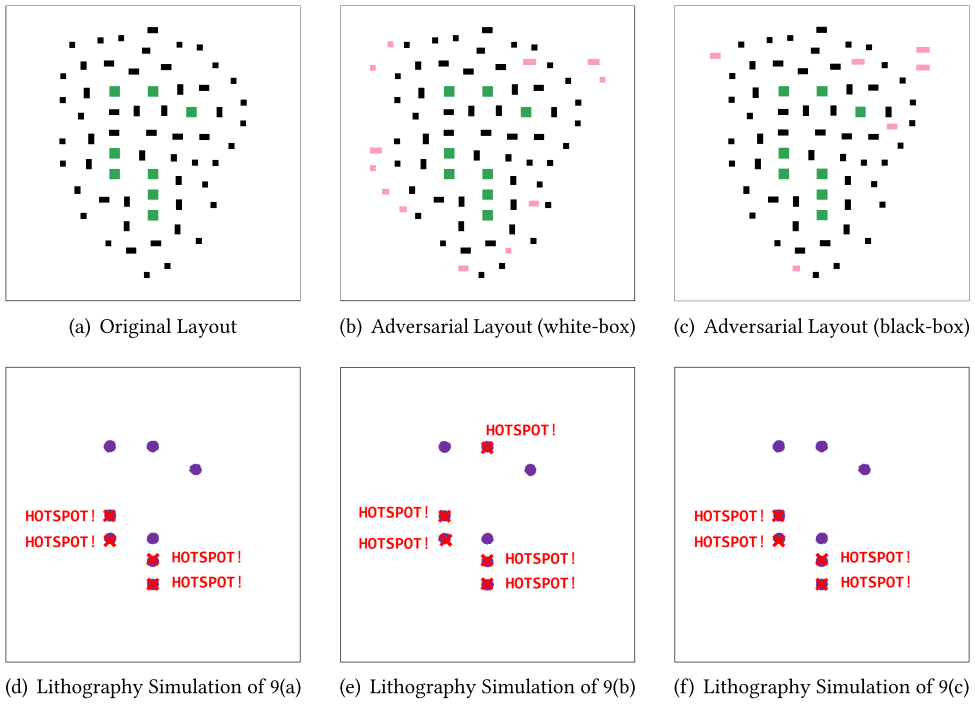
(d) Lithography Simulation of 8(a) (e) Lithography Simulation of 8(b) (f) Lithography Simulation of 8(c)

Fig. 8. White-box and black-box attack outputs—Example 1. These examples feature a single inserted SRAF. In the layout images (a–c), the vias are colored green, the original SRAFs are black, and the adversarial SRAFs are red. In the lithography simulation outputs, the vias are shown in purple and hotspots are marked with a cross and labelled with "HOTSPOT!"

Looking to the black-box attack results, we notice the same general trends as the white-box attack, where the simpler Network A is attacked most successfully, while the complex Network B exhibits greater robustness. The average attack time for the black-box attack is 350.5 s/677.3 s, for Networks A and B, respectively. Figures 7(b) and 7(d) show the percentage of layouts that were successfully perturbed by a given number of SRAF insertions for the black-box attack. As with the white-box attack, the black-box attack yielded adversarial layouts with as few as one inserted SRAF (an example is shown in Figure 8). Of the layouts that were successfully perturbed to appear as non-hotspot, ~18% required one inserted SRAF to fool Network A and ~11% required one SRAF to fool Network B. In the black-box attack on Network A, 50% of the adversarial layouts required three or fewer added SRAFs. For Network B, 6 or fewer SRAFs were required in 50% of the adversarial layouts.

6.3 Comparison and Observations

There are a number of commonalities and differences between the results of white-box and black-box attacks. Both attacks produced adversarial layouts with only one added SRAFs for a number of layouts. Attacks on the simpler Network A had a higher perturbation success rate in both white-box and black-box cases. There is also a notable increase in the average number of adversarial SRAFs added; the white-box attack requires 1–2 more SRAFs on average. An example of different SRAF insertions produced by the white-box and black-box attack can be seen in Figure 9.



(a) Original Layout (b) Adversarial Layout (white-box) (c) Adversarial Layout (black-box)

(d) Lithography Simulation of 9(a) (e) Lithography Simulation of 9(b) (f) Lithography Simulation of 9(c)

Fig. 9. White-box and black-box attack outputs—Example 2. These examples feature multiple inserted SRAFs. In the layout images (a–c), the vias are colored green, the original SRAFs are black, and the adversarial SRAFs are red. In the lithography simulation outputs, the vias are shown in purple and hotspots are marked with a cross and labelled with "HOTSPOT!"

While it may appear counter-intuitive that the black-box attack is able to produce adversarial examples with fewer SRAFs compared to the white-box approach (on average) given that the white-box approach uses more information about the targeted CNN, this is due to the greedy-search nature of the black-box attack, where, in each iteration, the perturbation (in a set of all valid perturbations) that makes the greatest change in output prediction is selected. In other words, the black-box approach finds the best perturbation to move the detector from classifying a clip as hotspot toward non-hotspot in each iteration of the black-box attack.

In contrast, the gradient-based insertion of the white-box attack does not guarantee that the best perturbation is chosen in each iteration. Instead, this additional information afforded to the white-box attacker is used to improve the speed of the attack. In the white-box attack, the gradients are calculated on a per-pixel level. The gradient represents how much the output prediction will change based on a *marginal* modification of the pixel value of the input, thus providing guidance as to where perturbations are *likely* to affect the output classification in the direction we desire (i.e., from hotspot to non-hotspot). Because SRAFs are large (i.e., they are a block of pixels), we choose, as candidates, the n number of *blocks* of pixels that, *together*, have the top-n largest negative gradient sums as candidates for perturbation. By only selecting n perturbation candidates, we reduce the number of times the attacker needs to query the CNN, thus speeding up the attack compared to the black-box attack. However, inserting an SRAF is a relatively large change (i.e., it is not a marginal modification); hence, there is no guarantee that a large change where the gradients are largest will result in the largest change in classification, as the gradient only represents the sensitivity to a *small* change. This is why we query the detector with each of the n candidates

in the white-box attack, and choose the best one out of these candidates. Also, this means the perturbation that will cause the largest change (as found by the black-box attack) could be outside the top-n candidates found by the white-box attack. That is why, on average, white-box attacks may require more perturbations to mislead the detector from hotspot to non-hotspot.

Of particular interest to an attacker is the feasibility of the attack in terms of computational overhead. One measure of this is the time taken to generate an adversarial layout. Our results show that the more successful black-box attack is 10× slower than the white-box attack. The trade-off for such high attack success is increased time and computation requirements. This can be explained by the number of times the attacker needs to query the CNN-based detector. In the white-box attack, the attacker only needs to query the network $n + 1$ times in each iteration. The first query is incurred when using the network to compute the loss function gradients for each pixel. The subsequent n queries obtain the prediction probabilities for candidate adversarial layouts. When n is set to the maximum number of possible shape/position combinations, white-box is equivalent to the black-box attack.

As an attacker, one could tune the *check parameter*, n, to balance the perturbation success rate against the computation resources required. Thanks to the gradient information used to guide the placement of adversarial SRAFs in the white-box algorithm, n need not be too large to achieve reasonable or even comparable perturbation success rate as the black-box attack, while taking advantage of up to a 10× attack time reduction. This can also be useful for defenders, as we discuss in Section 7.

6.4 Do Adversarial Perturbations Fix Hotspots?

Given the success of our adversarial perturbations, a natural question to ask is whether, using perturbations, we are actually fixing hotspots instead of misleading the designer. To answer this question, we performed lithography simulation of the adversarial layouts to confirm the hypothesis that inserting only a few SRAFs does not drastically improve/fix hotspots but instead cause misclassification in most cases. We used the same experimental settings as those for ascertaining the ground truth labels of the original dataset (described in Section 4). Examples of original and adversarial layouts, as well their simulation outputs, are shown in Figures 8 and 9. The simulations revealed that in the majority of cases, our adversarial layouts still produced layout defects. In the white-box attack on Network A, 84.4% of the adversarial layouts that the network classified as non-hotspot were verified as hotspot, while in the same attack on Network B, 77.4% of the adversarial layouts that were classified as non-hotspot were verified as hotspot. Based on this, we can consider the *true attack success rate*, being the percentage of perturbed hotspots that remain as hotspots but *misclassified* as non-hotspots, as ∼84% and ∼66%, respectively. In the black-box attack, the hotspot verification rate was similar, where 86.7% and 77.9% of the layouts that fooled Network A and B (respectively) were verified as hotspot corresponding to a true attack success rate of ∼86% and ∼73%, respectively. The lower true attack success rate on Network B suggests that the deeper network is more robust compared to Network A, despite their apparently similar baseline accuracy. This implies that accuracy alone is not a determining factor for robustness against adversarial perturbations, and we further discuss this notion in Section 8. In the case for both networks, this level of misclassification under an adversarial setting nevertheless raises concerns about the overall robustness of such CNN-based hotspot detectors.

Additionally, when we examined the lithography simulation outputs, we found instances where the number of hotspots in a layout increased, decreased, and stayed the same. An example of an instance where the inserted SRAFs added hotspots is shown in Figure 9(e), instances where the inserted SRAFs led to less hotspots is shown in Figures 8(e) and 8(f), and an instance where the inserted SRAFs did not change the hotspot number is shown in Figure 9(f).

ALGORITHM 3: Adversarial Retraining

Input: Training data D_{train}, Training hotspot data $D_{train,hotspot}$, network function F, adversarial non-hotspot layout generation function $Attack$, lithography simulation process $Litho$, network retraining process $Retrain$, maximum number of retraining rounds R.
for i = 1 to R **do**
    $adv\_train = Attack(F, D_{train,hotspot})$ ▷ Generate adversarial non-hotspot layout for training hotspot.
    $adv\_train\_hotspot = Litho(adv\_train)$ ▷ Get verified hotspot through lithography simulation.
    $D_{train} = D_{train} \cup adv\_train\_hotspot$
    $F = Retrain(F, D_{train})$ ▷ Retrain network F with robust training data D_{train}.
Return: Robustified network F

7 TOWARD A MORE ROBUST NETWORK

7.1 Iterative Adversarial Retraining

So far, we have shown that the white-box and black-box attacks are effective. Since this implies a feasible threat to deep learning-based CAD, there is a need to investigate and propose countermeasures. As such, we propose a strategy to increase the robustness of CNN-based hotspot detectors. The main aim is to reduce the perturbation success rate without compromising hotspot detection accuracy. The approach can be integrated into the initial training process for the CNN-based network and is a type of adversarial retraining, as proposed in Reference [47].

First, let us assume that the defender knows the risks of adversarial perturbations on hotspot detectors. Intuitively they can make the trained network robust by including adversarial layouts into the training dataset but with true hotspot labels, and then retrain their detector using the usual methods [47]. To diversify the training data set, the defender can adopt the attacker's methodology to proactively generate their own adversarial layouts and include them after verifying the true labels using lithography simulation. For robustness, the defender can repeat the adversarial retraining to suppress the success rate of adversarial attacks on the robust retrained network.

In practice, as we showed in Section 6, the black-box attack achieves the highest possible perturbation success rate. Hence, it would make sense to make the network robust using adversarial layouts produced by this attack. However, given that it can be 10× slower than the white-box attack, this is less feasible under time and computation resource constraints. Therefore, while the white-box attack may have lower success rate in some occasions, it is more efficient in generating adversarial layouts. We adopt the white-box attack as part of the defender's strategy, and this provides ample training data and robustification results. Adversarial retraining is shown in Algorithm 3 and the flow is shown in Figure 10.

7.2 Evaluation

To demonstrate robustification, we perform iterative adversarial retraining on Network A, and perform white-box attacks to determine the perturbation success rate. We start with Network A in Section 5. We conduct the white-box attack using all hotspot layouts from the training set that are correctly classified by the CNN (① of Figure 10). Of these 2,070 hotspot layouts, the white-box attack successfully produces 1,725 adversarial layouts (these were verified using lithography simulation, ② of Figure 10). These are labelled as hotspot, and added into the training dataset for the 1st round of retraining (③ of Figure 10). We call the 1st round retrained Network A′. For the second round of retraining, we take the hotspot layouts from the expanded training dataset that are correctly classified by Network A′, and perform the white-box attack. A′ classifies 3,910 hotspot layouts correctly, and from these, the white-box attack produces 2,141 lithography simulation-

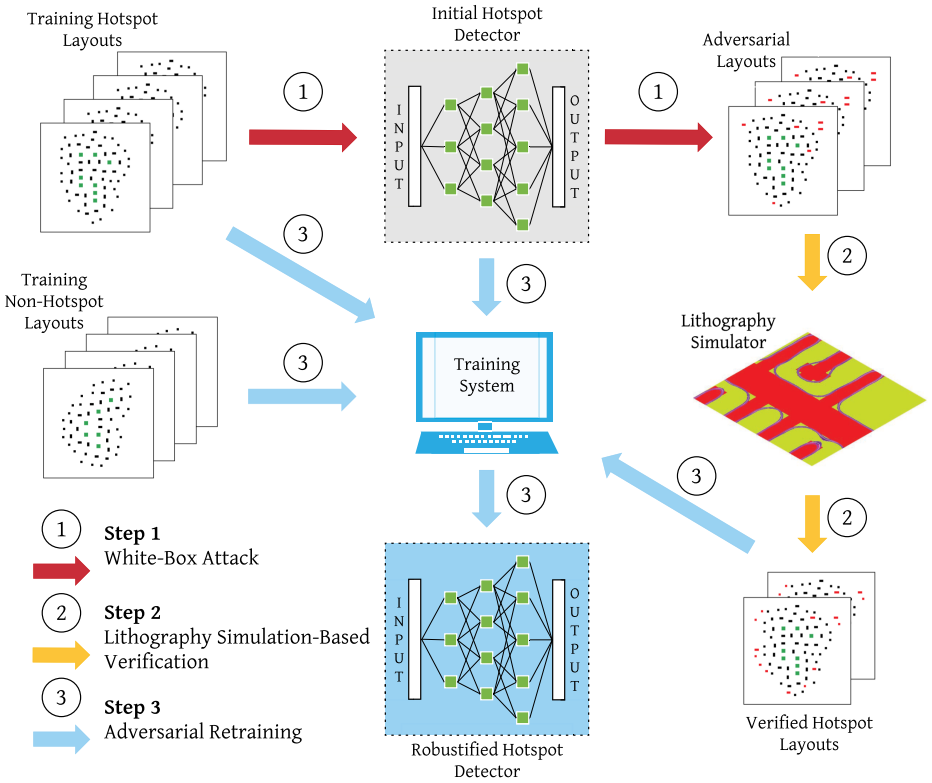


Fig. 10. Overview of the adversarial retraining process.

Table 5. White-box Attack Results with Iterative Adversarial Retraining on Network A

Network	Initial net (A)	First round retrain (A′)	Second round retrain (A″)
Network overall accuracy	0.73	0.73	0.73
Hotspot detection accuracy	0.72	0.72	0.72
perturbation success rate	99.7%	73.6%	37.2%
Average attack time per layout	8.6 s	18.2 s	22.9 s
Average number of SRAFs added	5.3	7.3	7.2
Average area of SRAFs added	0.3%	0.4%	0.4%

The maximum number of SRAF insertions allowed (T) is 20, and the check parameter (n) is 180.

verified hotspot layouts. These layouts are added into the training dataset, and we perform a second round of retraining to produce the next generation of robustified network, Network A″. One can repeat this until an attack threshold success rate is met, or after a pre-determined number of rounds.

To evaluate the efficacy of adversarial retraining, we perform white-box attacks using correctly classified hotspot layouts for Network A, A′, and A″ from the validation dataset, after each round of training. When we use the white-box attacks to produce new training data and evaluate the retrained networks, we set the maximum number of SRAF insertions allowed (T) to 20, and the check parameter (n) to 180 (as in Section 6). The results are shown in Table 5. Throughout the

retraining process, the networks' overall accuracy (the average of hotspot and non-hotspot classification accuracy) and hotspot detection accuracy are maintained. Even though Network A has a simpler architecture, after two rounds of retraining, its robustness to being fooled by adversarial input perturbation (as represented by the decreased perturbation success rate) surpasses that of Network B (where the perturbation success rate was 85.5%).

8 DISCUSSION AND ADDITIONAL INSIGHTS

Over the course of this study, our findings raised several questions that warrant discussion and future study.

What drives differences between white-box vs. black-box perturbation success rate? Black-box attacks had a higher success rate compared to white-box attacks, at the cost of longer attack time. We posit that this is largely due to the greedy nature of the black-box attack, where the target network is repeatedly queried. This query-based approach guarantees that the attacker can achieve the highest perturbation success rate (given a fixed horizontal and vertical sliding stride) in searching for the best shape/position combination. Conversely, while the white-box attack is a gradient-guided approach, it considers n candidates and it is possible that the best valid solutions are missed depending on the size of the check parameter.

What factors affect differences in robustness between different networks? An interesting finding is that that there was a difference in perturbation success rate and true attack success rate against Network A and B, where the more complex Network B displayed greater robustness even while the networks' baseline hotspot detection accuracies were the same. Prior work has found that networks with greater capacity are more robust [29]; whether Network B has learned a "better" approximation of the underlying physics warrants more study. As mentioned in Section 4.2, the hotspot detectors we trained did not appear to be overfitted to the training data (as there was less than 3% difference between training and test data accuracy in both Network A and B). Taken together, this suggests that neither baseline accuracy nor overfitting act as the determining factor for robustness; this is also echoed in recent studies [48, 61]. In our adversarial retraining process, we found that robustness was improved without affecting accuracy (i.e., neither improving nor degrading the hotspot/non-hotspot classification accuracy), so the perturbed layouts do not appear to provide sufficient additional information to improve generalizability, but nevertheless enable improved robustness to adversarial input perturbation. We also wanted to see if the number of DCT coefficients used in the input layer affects a detector's vulnerability to adversarial input perturbation (after all, Network B uses more DCT coefficients in its input). To do this, we trained another CNN (using the same architecture as Network A), which instead used 160 DCT coefficients as its input (i.e., 5× the number of coefficients compared to the input layer of Network A), to see if the CNN would learn more complex details from the input features, thus improving robustness. We performed the white-box attack on this new network, using the same settings described in Section 5, and achieved a 100.0% perturbation success rate, where the average number of SRAFs added was 2.9 (or +0.2% area). This suggests that using only a few of the lower frequency DCT coefficients was not the root-cause of the vulnerability to adversarial input perturbation.

What is the impact of design rules on the ability to produce adversarial examples? Unlike research into adversarial input perturbation in general image classification settings (e.g., Reference [14]), we focused on the notion of *semantically meaningful perturbations*. In the CAD context, this refers to manipulation of design artifacts while satisfying design rules. Given more complex design rules, the space for valid perturbations becomes smaller. While this may speed up the process of searching for perturbations (as there are less options to explore), it may also increase the difficulty of

finding the required perturbations while maintaining some desired measure of imperceptibility or innocuousness. Depending on an adversary's goal, a single act of sabotage may be sufficient to achieve a desired malicious derailment. Going forward, future study could explore the interplay of complex design rules at more advanced nodes, and protection against meaningful perturbations, thus reducing the incidence of such error-causing corner cases.

Do adversarial attacks generalize to other datasets? Our study focuses on SRAF insertion for improving the printability of via layouts, which represents only one scenario in lithographic hotspot detection. Unfortunately, as noted previously, we were limited in our ability to comprehensively evaluate our attack on the ICCAD'12 contest benchmarks due to the unavailability of a DRC deck and lithography simulation settings for these benchmarks. Nevertheless, in the appendix, we show results for a limited evaluation of adversarial perturbation attacks on the ICCAD'12 dataset using a small set of inferred design rules. In this experiment, we show that our attack strategies are indeed able to produce adversarial examples. Note that although we could not verify via lithography simulation that the adversarially perturbed layouts remain hotspots, the perturbations are relatively small and are therefore unlikely to have actually fixed hotspots (as we have observed in the SRAF case study). Interestingly, the baseline CNN for the ICCAD'12 benchmarks has >90% accuracy, suggesting that higher accuracy by itself does not necessarily imply immunity to adversarial perturbations. The perturbation success rate in this experiment is somewhat lower than the results of Section 6, perhaps suggesting that the ease of finding adversarial examples is affected, at the very least, by the nature of the dataset and the interplay between design rules and the chosen network architecture.

Do adversarial perturbations transfer across detectors? In our experiments, we focused on CNN-based hotspot detectors, demonstrating their vulnerability to adversarial input perturbation. An interesting phenomenon that has been observed in adversarial machine learning research on image classification has raised the possibility of perturbations that are able to fool DNN models with different architectures [35]. To investigate whether this may also be the case in our setting, we trained another CNN-based hotspot detector, instead with 13 layers (sitting between the 9-layer Network A and 15-layer Network B). This 13-layer network is similar to Network A but with two additional convolution layers before each maxpooling layer. We took the successfully perturbed clips from the white-box attack on Network A, and used Network B and the 13-layer CNN to classify them. None of the perturbed clips that fool Network A fool Network B, but 10.5% of them fool the 13-layer CNN. Seeing as all the networks have essentially the same baseline accuracy, the poor transferability of perturbed clips appear to hint at the different networks' learning of different features for classification, suggesting that generated adversarial examples appear to be specific to a single detector, although this could be investigated more thoroughly in future work.

It is also worth discussing whether adversarial examples might also affect other hotspot detection approaches, such as those involving pattern matching. Pattern matching-based approaches compare layouts against a database of known (previously-seen) hotspot-susceptible patterns. Because adversarial input perturbation processes are designed to minimize the modifications made to a clip, pattern matching approaches that are fuzzy (e.g., Reference [49]) are more likely to identify the subtle changes as simply variants of hotspot patterns (assuming, of course, that the hotspot pattern is in the database of known patterns). If an "exact pattern" identification approach is used, however, then perturbations could indeed fool the detector by making the clip sufficiently vary from known clips in the database.

Does adversarial retraining using one attack algorithm protect against all possible adversarial insertions? Different attack formulations may discover similar and different adversarial examples—we

witnessed this situation in our exploration of the white-box and black-box attacks. We expect that retraining on adversarial examples from one type of perturbation-based attack (e.g., our white-box attack) would provide some robustness against other types of perturbation-based attack (e.g., our black-box attack). However, we cannot guarantee that an adversarially retrained network (using adversarial samples from one attack algorithm) is robust against all other attack. This is because the perturbation space could be large, and the other attack algorithms may find non-overlapping samples from this space. Thus, we expect that networks robustified using adversarial retraining are robust to some, but not all adversarial insertions—this poses another interesting line of inquiry for future work.

What does the network learn? Security aside, perhaps the more basic question raised by our work is this: what does a CNN learn? The high success rate of the attacks indicate that the CNN-based hotspot detectors do not truly and fully "learn" the physics relevant to the hotspot problem. One might argue that this is to be expected; after all, a CNN is only approximating the underlying physics. Nonetheless, the fact that in several cases only one or two additional SRAFs throws off the CNN is worrisome, since, at least intuitively, these small modifications should not drastically fix hotspots. Furthermore, the ability to acquire or prepare a "perfect" dataset that fully represents the complete domain space of a problem, and the ability to guarantee that training fully "learns" this domain space remains an open research problem, so there is a pressing need to understand what a DNN learns in imperfect settings. Indeed, in the broader deep learning community, there is on-going work about the interpretability of neural networks that seeks to better understand what concepts networks actually learn, and we would encourage this to be considered in the ML-CAD context also [33].

Are there wider security implications for ML-based CAD flows? The success of both attacks in lithographic hotspot detection raises important questions about the wider implications to ML in CAD. CAD flows involve many complex steps using tools sourced from different vendors; this provides a wide attack surface [2]. With ML added to the mix, the risk compounds—prior work has raised concerns about outsourcing deep learning [45], and given the many CAD domains in which deep learning can contribute (we provide an overview in Section 9), security considerations are paramount. A key insight we provide in this study is the presence of semantically meaningful perturbations in the lithographic context. We posit that similar meaningful perturbations exist in other CAD domains, and further work should be done to discover these. Moreover, we contend that the data that is produced and used for training in CAD problems is imperfect (i.e., that it is infeasible to generate data that completely captures the full gamut of possible designs and processes). Given this, we surmise that *any* current ML-CAD solution using neural networks is therefore imperfect to some degree, and thus, could in fact be a potential target.

9 RELATED WORK

The CAD industry is facing challenges with increasing design complexity, especially with the growing time-to-market pressures. ML techniques have been explored to accelerate steps in the VLSI design flow [20].

In optical lithography, a variety of techniques have been proposed to analyze the printability of layouts and design enhancement. Pattern matching (such as Reference [60]) and ML (such as Reference [30]) have been studied, offering a range of successes in accuracy and ability to generalize for previously unseen layouts. Recently, there has been an uptick in the study of deep learning approaches, where different facets have been investigated. For example, in Reference [56], Yang et al. train a CNN to detect hotspots from a layout image. They provide a detailed comparison on the effectiveness of different ML techniques at identifying hotspots, concluding

that the CNN-based approaches offer superior accuracy. To reduce the computational overhead of processing large layout data, Reference [19] proposes to binarize and down-sample the input data, yielding 8× speed-up over prior deep learning solutions.

Other studies have exposed challenges in adopting CNNs, such as the abundance (or lack thereof) of labeled data for training. Chen et al. detect hotspots using a CNN [7], but propose a semi-supervised approach to handle the scarcity of labelled data. Using a two-stream architecture, labelled data is used to create a preliminary model that is used to provisionally label other samples together with a measure of confidence in the provisional label. Provisionally labelled samples with high confidence are used to train the model in subsequent training cycles; the general belief is that more data can result in ML models with more knowledge. Synthetic variants of labelled data are proposed in Reference [39] to improve the information-theoretic content of the training dataset and address dataset imbalances, resulting in considerable improvement of false alarm rates. The adversarial retraining procedure proposed in our work can similarly be viewed as a data augmentation strategy; however, unlike prior work, our data augmentation is targeted toward generating adversarially robust networks.

Recently, state-of-the-art applications of CNNs have moved beyond design analysis toward design *enhancement* to aid in modifying designs to reach a certain goal. Insertion of SRAFs has been framed as a type of image domain transformation, where Generative Adversarial Networks (GANs) are trained to take in layouts and "predict" where SRAFs should be inserted [1]. Other mask optimizations (such as OPC) have been cast similarly [13, 53, 54, 58]. While they focus on accuracy and scalability, our work examines an orthogonal, yet crucial dimension of robustness. In physical design, trained ML models are a faster alternative to simulation, allowing designers to quickly evaluate the validity of a design. Lin et al. perform resist modelling and demonstrate transfer learning for different technologies [25]. Cao et al. [5] use parameters related to design, pin-mapping, and layout to predict achievable and actual inductance at pre- and post-layout stages.

Checking design rule violation (DRV) is another important aspect of the design flow where deep learning has been used. In Reference [44], Tabrizi et al. do routability checks after netlist placement, but before global routing. Routing shorts are predicted using the trained model, allowing designers to avoid potential unroutable layouts. Similarly, Xie et al. [51] use a CNN to predict the number of DRVs, even in the presence of design macros and to identify DRV hotspots. DRV prediction ascertains the routability of layouts for earlier correction.

In early stages of design, deep learning has been used for logic optimization [50], design space exploration [16], synthesis flow exploration [59], and high-level area estimations [11]. Such techniques reduce designer workload by culling the design variants that need to be progressed in the design flow. Yu et al. [59] propose the training and use of a CNN to gauge the effectiveness of different combinations of synthesis transformations (termed *flows*) for a given register transfer level (RTL) design. To avoid exhaustively running all combinations, a model is trained from a *subset* of possible flows. The trained model is used to predict the quality of several possible flows, outputting a collection of "angel-flows" that are likely to yield good results in the synthesis of the RTL design. In a related approach, Hasswijk et al. [50] train a CNN to discover new transformation algorithms, but focus on graph optimization instead of quality of results from subsequent technology mapping. Orthogonal to these approaches, Greathouse et al. [16] use a neural network to predict how the performance of a software kernel will scale as a function of the number of parallel compute units. The adversarial robustness of these aforementioned approaches remains an open question.

Adversarial attacks on CNNs are being actively studied [3], although, prior to our article, this question of adversarial robustness has not yet been brought up in the electronic CAD domain. Our work contributes to this growing body of literature in this hitherto unanalyzed context. Our attacks modify layouts to cause misclassification on a well-trained network; these belong to the

class of *inference-time* or *evasion* attacks and have been examined in detail (in a general context) in works like [12, 14, 24, 35, 42, 43].

Adversarial attacks in the literature can be classified into two categories based on the contextual meaning/imperceptibility of the added perturbation. One class of adversarial perturbations are meaningless and akin to subtle noise that are crafted to fool the neural network in general cases. The other type of adversarial perturbations have contextual meaning, while still remaining subtle in the semantics of the real-world context. Examples of these include using a pair of glasses to mislead a face recognition system [42] and a post-it note on a traffic sign to fool a traffic sign detector [12]. Our attack on hotspot detection falls into the second category, as our added SRAFs are semantically meaningful (SRAFs are real-world artifacts) and difficult to perceive as malign.

A variety of defenses against adversarial inputs have been proposed [10, 29, 37, 47]. Some focus on the detection of adversarial inputs by identifying feature disparity between valid and adversarial examples [31, 32, 52]. Some resort to transformation techniques to rectify adversarial inputs into "normal" ones [17, 31, 41]. Others resort to retraining to counter adversarial attacks [14, 47]. Research into defenses that offer strong guarantees of robustness is ongoing [29]. We use adversarial retraining to make CNN-based hotspot detectors robust without sacrificing detection accuracy and adding computation overhead in inference.

In contrast to *inference-time* attacks, another class of attacks on deep learning is that of *training-time* or *backdoor* attacks [28, 45]. Here the training dataset is in some way compromised (or poisoned). Part of the study into these attacks includes examining the risks when the integrity of the training data is compromised [45], and the risks that come from re-using potentially compromised networks. Recent work that aims to improve the robustness to backdoors include [4, 6, 26]. Understanding the implications of these attacks in ML-based CAD merits investigation.

10 CONCLUSION

In this article, we revealed a vulnerability of CNN-based hotspot detection in electronic CAD. We showed that CNN-based hotspot detectors are easily fooled by specially crafted SRAF insertions that can mislead the network to predict a hotspot layout as non-hotspot. We proposed and examined white-box and black-box attacks on well-trained hotspot detection CNNs, and the results showed that up to 99.7% perturbation success rate was possible. The deeper, more complex CNN we attacked exhibited better natural robustness compared to the less complex CNN. To robustify the vulnerable hotspot detectors, we proposed adversarial retraining, revealing that after only two rounds, the white-box perturbation success rate could be decreased to 37.2%. Our findings point to semantically meaningful adversarial perturbations as a viable concern for ML-based CAD. This study leads us to urge caution and advocate for further study of the wider security implications of deep learning in this field. As an immediate recommendation for CNN-based hotspot detection, we suggest adversarial retraining as an add-on procedure after initial network training, as it introduces no extra computation overhead at inference and has no accuracy compromise, but adds robustness against adversarial attacks. Ultimately, we find that adversarial perturbations should be a concern for ML-based CAD, and thus advocate that an appropriately proactive stance is adopted as such systems mature. Hence, our future work will look to other attack types in other CAD problems, including training-time attacks and robustification techniques.

APPENDIX

EXPLORATION USING ICCAD'12 DATASET

To further explore wider security implications in ML for CAD, we investigated adversarial perturbations in another hotspot detection task. In this experiment, we attacked a different CNN-based

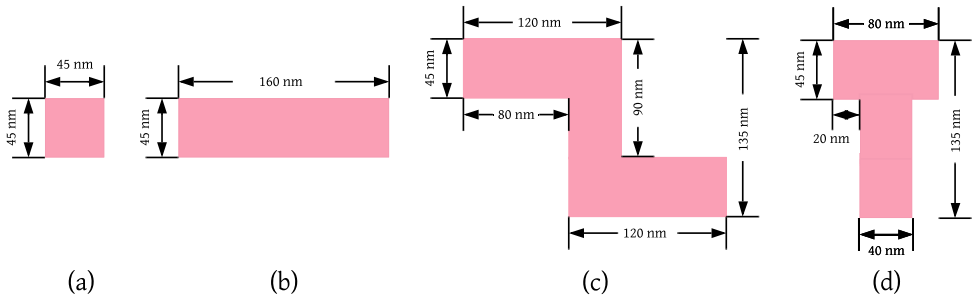


Fig. 11. Restricted set of patterns for black-box attack (shape dimensions shown). The (a) square, (b) rectangle, (c) zigzag, and (d) Tee shapes are drawn from the existing layout dataset.

hotspot detector, trained using a dataset from the ICCAD'12 contest on pattern matching for physical verification [46]. The attack goal is similar to that in Section 2 in that the attacker wants to modify given hotspot layouts such that they are misclassified as non-hotspot by the detector. Note, however, we could not perform lithography simulation-based verification as the physical models required were not available as part of the ICCAD competition dataset.

Dataset and Hotspot Detector Design. We use layout 4 from the ICCAD dataset containing 4,547 training and 32,067 test layout images; 4,452 of the training samples are non-hotspot and the remaining 95 are hotspot. Of the test samples, 31,890 are non-hotspot and 177 are hotspot. Each layout image has dimensions of 1,200 × 1,200 pixel, and has binary valued pixel intensities to represent the pattern to be printed. Each pixel corresponds to $1nm^2$ of the layout. Before training and inference, we preprocess the layout images using the same DCT filters as described in Section 4.2 to obtain the DCT coefficients as input to the hotspot detector. The resulting input dimension is (12, 12, 32). We use a similar CNN architecture and training procedure as for Network A (Section 5) but with this modified input dimension — the network parameters are shown in Table 6. The trained network has 98.5% non-hotspot classification accuracy and 92.7% for hotspot classification accuracy on the test data.

Table 6. Network Architecture

Layer	Kernel Size	Stride	Output Size
input	-	-	(12, 12, 32)
conv1_1	3	1	(12, 12, 16)
conv1_2	3	1	(12, 12, 16)
maxpooling1	2	2	(6, 6, 16)
conv2_1	3	1	(6, 6, 32)
conv2_2	3	1	(6, 6, 32)
maxpooling2	2	2	(3, 3, 32)
fc1	-	-	250
fc2	-	-	2

Black-box attack. We conducted a black-box attack on the test hotspot layouts to examine the efficacy of our proposed attack scheme. However, instead of inserting SRAFs, we add isolated printing patterns to the layouts. The attack constraints in this experiment are: (1) shape constraint: adversarial insertions can only be chosen from a restricted set of four basic shapes that already exist in the layout dataset, as illustrated in Figure 11; (2) spacing constraint: inserted patterns

Table 7. Summary of Black-box Attack Result

Perturbation success rate	77.4%
Average attack time per layout	63.5 s
Average number of patterns added	4.5
Average area of patterns added	2.1%

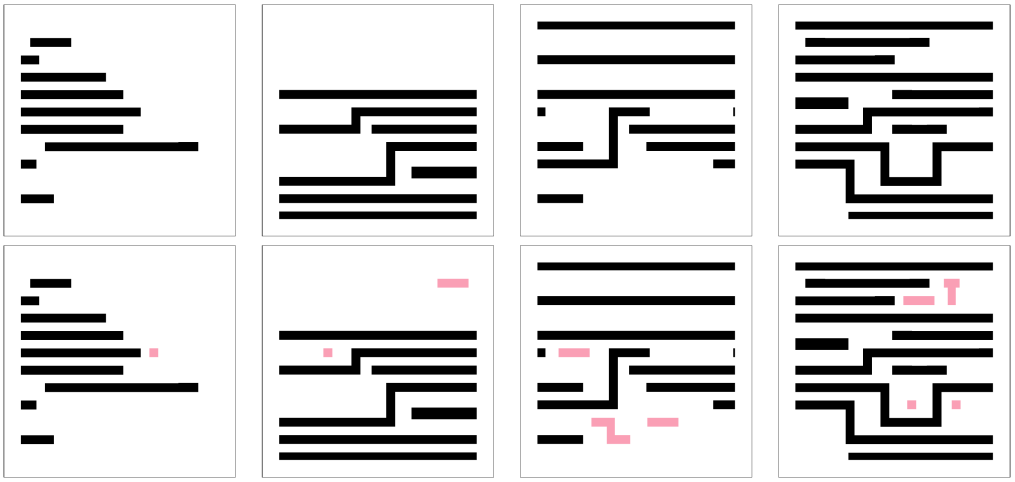


Fig. 12. Top row: original hotspot layouts. Bottom row: corresponding adversarial non-hotspot layouts.

should be at least 45 nm away from any surrounding patterns; (3) alignment constraint: inserted patterns need to be aligned with existing shapes; (4) insertion region: inserted patterns must not overlap with a 100-nm-wide border at the edges of the layout image. The black-box algorithm is reused from Section 5.

Attack Results. We performed the black-box attack using 164 hotspot layouts from the test set that the detector correctly classified as hotspot. Of these 164 layouts, 127 were successfully perturbed to fool the network. The results are summarized in Table 7. We illustrate some of the perturbed layouts in Figure 12 with 1–4 adversarial inserted patterns.

Remarks. Based on these additional experiments, it appears that this dataset is also susceptible to adversarial perturbation attacks, despite the fact that the CNN-based hotspot detector baseline accuracy on this dataset is high. Access to the lithography simulation settings will further allow us to verify that the modified layouts remain hotspots.

REFERENCES

[1] Mohamed Baker Alawieh, Yibo Lin, Zaiwei Zhang, Meng Li, Qixing Huang, and David Z. Pan. 2019. GAN-SRAF: Sub-Resolution Assist Feature Generation Using Conditional Generative Adversarial Networks. In *Proceedings of the Design Automation Conference (DAC'19)*. ACM, 1–6. DOI: https://doi.org/10.1145/3316781.3317832

[2] Kanad Basu, Samah Mohamed Saeed, Christian Pilato, Mohammed Ashraf, Mohammed Thari Nabeel, Krishnendu Chakrabarty, and Ramesh Karri. 2019. CAD-Base: An Attack Vector into the Electronics Supply Chain. *ACM Trans. Des. Autom. Electron. Syst.* 24, 4 (Apr. 2019), 38:1–38:30. DOI: https://doi.org/10.1145/3315574

[3] Battista Biggio and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recogn.* 84 (Dec. 2018), 317–331. DOI: https://doi.org/10.1016/j.patcog.2018.07.023

[4] Bolun Wang, Yuanshun Yao, Shawn Shan, Huiying Li, Bimal Viswanath, Haitao Zhen, and Ben Y. Zhao. 2019. Stealthy porn: Understanding real-world adversarial images for illicit online promotion. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'19)*, Vol. 1. IEEE Computer Society, 547–561. DOI: https://doi.org/10.1109/SP.2019.00032

[5] Yi Cao, Andrew B. Kahng, Joseph Li, Abinash Roy, Vaishnav Srinivas, and Bangqi Xu. 2019. Learning-based prediction of package power delivery network quality. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'19)*. ACM, 160–166. DOI: https://doi.org/10.1145/3287624.3287689

[6] Bryant Chen, Wilka Carvalho, Nathalie Baracaldo, Benjamin Edwards, Taesung Lee, Heiko Ludwig, Ian Molloy, and Biplav Srivastava. 2019. Detecting backdoor attacks on deep neural networks by activation clustering. In *Proceedings of the AAAI Workshop on Artificial Intelligence Safety (SafeAI'19)*. AAAI, 8.

[7] Ying Chen, Yibo Lin, Tianyang Gai, Yajuan Su, Yayi Wei, and David Z. Pan. 2019. Semi-supervised hotspot detection with self-paced multi-task learning. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'19)*. ACM, 420–425. DOI: https://doi.org/10.1145/3287624.3287685

[8] François Chollet et al. 2015. Keras. Retrieved from https://keras.io.

[9] Calma Company. 1987. GDSII Stream Format Manual, Release 6.0. Retrieved from http://bitsavers.informatik.uni-stuttgart.de/pdf/calma/GDS_II_Stream_Format_Manual_6.0_Feb87.pdf.

[10] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. 2018. Stochastic activation pruning for robust adversarial defense. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*. OpenReview.net, 1–6. Retrieved from https://openreview.net/forum?id=H1uR4GZRZ.

[11] Elena Zennaro, Lorenzo Servadei, Keerthikumara Devarajegowda, and Wolfgang Ecker. 2018. A machine learning approach for area prediction of hardware designs from abstract specifications. In *Proceedings of the 21st Euromicro Conference on Digital System Design (DSD'18)*. IEEE Computer Society, 413–420. DOI: https://doi.org/10.1109/DSD.2018.00076

[12] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'18)*. IEEE Computer Society, 1625–1634. DOI: https://doi.org/10.1109/CVPR.2018.00175

[13] Hao Geng, Haoyu Yang, Yuzhe Ma, Joydeep Mitra, and Bei Yu. 2019. SRAF insertion via supervised dictionary learning. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'19)*. ACM, New York, NY, 406–411. DOI: https://doi.org/10.1145/3287624.3287684

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*. Retrieved from http://arxiv.org/abs/1412.6572.

[15] Mentor Graphics. 2019. Calibre LFD. Retrieved from https://www.mentor.com/products/ic_nanometer_design/design-for-manufacturing/calibre-lfd/.

[16] Joseph L. Greathouse and Gabriel H. Loh. 2018. Machine learning for performance and power modeling of heterogeneous systems. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'18)*. ACM, 1–6. DOI: https://doi.org/10.1145/3240765.3243484

[17] Chuan Guo, Mayank Rana, Moustapha Cissé, and Laurens van der Maaten. 2018. Countering adversarial images using input transformations. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=SyJ7ClWCb.

[18] Haibo He and Edwardo A. Garcia. 2008. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*9 (2008), 1263–1284. DOI: https://doi.org/10.1109/TKDE.2008.239

[19] Yiyang Jiang, Fan Yang, Hengliang Zhu, Bei Yu, Dian Zhou, and Xuan Zeng. 2019. Efficient layout hotspot detection via binarized residual neural network. In *Proceedings of the Design Automation Conference (DAC'19)*. ACM, 1–6. DOI: https://doi.org/10.1145/3316781.3317811

[20] Andrew B. Kahng. 2018. Machine learning applications in physical design: Recent results and directions. In *Proceedings of the International Symposium on Physical Design (ISPD'18)*. ACM, 68–73. DOI: https://doi.org/10.1145/3177540.3177554

[21] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Retrieved from http://arxiv.org/abs/1412.6980.

[22] Jesper Knudsen. 2008. Nangate 45nm open cell library. *CDNLive, EMEA* (2008).

[23] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. Citeseer.

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *5th International Conference on Learning Representations (ICLR'17), Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net. Retrieved from https://openreview.net/forum?id=HJGU3Rodl.

[25] Yibo Lin, Yuki Watanabe, Taiki Kimura, Tetsuaki Matsunawa, Shigeki Nojima, Meng Li, and David Z. Pan. 2018. Data efficient lithography modeling with residual neural networks and transfer learning. In *Proceedings of the International Symposium on Physical Design (ISPD'18)*. ACM, 82–89. DOI: https://doi.org/10.1145/3177540.3178242

[26] Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2018. Fine-Pruning: Defending against backdooring attacks on deep neural networks. In *Research in Attacks, Intrusions, and Defenses (Lecture Notes in Computer Science)*, Michael

Bailey, Thorsten Holz, Manolis Stamatogiannakis, and Sotiris Ioannidis (Eds.). Springer International Publishing, 273–294. DOI: https://doi.org/10.1007/978-3-030-00470-5_13

[27] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. 2017. Delving into transferable adversarial examples and black-box attacks. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=Sys6GJqxl.

[28] Yingqi Liu, Shiqing Ma, Yousra Aafer, Wen-Chuan Lee, Juan Zhai, Weihang Wang, and Xiangyu Zhang. 2018. Trojaning attack on neural networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'18)*. The Internet Society. Retrieved from http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-5_Liu_paper.pdf.

[29] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. Retrieved from http://arxiv.org/abs/1706.06083.

[30] Tetsuaki Matsunawa, Jhih-Rong Gao, Bei Yu, and David Z Pan. 2015. A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction. In *Design-Process-Technology Co-optimization for Manufacturability IX*, Vol. 9427. International Society for Optics and Photonics, 94270S. DOI: https://doi.org/10.1117/12.2085790

[31] Dongyu Meng and Hao Chen. 2017. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 135–147. DOI: https://doi.org/10.1145/3133956.3134057

[32] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. 2017. On detecting adversarial perturbations. In *Proceedings of the 5th International Conference on Learning Representations (ICLR'17)*. OpenReview.net. Retrieved from https://openreview.net/forum?id=SJzCSf9xg.

[33] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Process.* 73 (2018), 1–15. DOI: https://doi.org/10.1016/j.dsp.2017.10.011

[34] Samuel K. Moore. 2018. DARPA Picks Its First Set of Winners in Electronics Resurgence Initiative. Retrieved from https://spectrum.ieee.org/tech-talk/semiconductors/design/darpa-picks-its-first-set-of-winners-in-electronics-resurgence-initiative.

[35] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. 2017. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'17)*. IEEE Computer Society, 86–94. DOI: https://doi.org/10.1109/CVPR.2017.17

[36] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on International Conference on Machine Learning (ICML'10)*. Omnipress, 807–814. Retrieved from http://dl.acm.org/citation.cfm?id=3104322.3104425.

[37] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *Proceedings of the IEEE Symposium on Security and Privacy (SP'16)*. 582–597. DOI: https://doi.org/10.1109/SP.2016.41

[38] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In *Proceedings of the ACM Asia Conference on Computer and Communications Security (ASIACCS'17)*. ACM, 506–519. DOI: https://doi.org/10.1145/3052973.3053009

[39] Gaurav Rajavendra Reddy, Constantinos Xanthopoulos, and Yiorgos Makris. 2018. Enhanced hotspot detection through synthetic pattern generation and design of experiments. In *Proceedings of the IEEE 36th VLSI Test Symposium (VTS'18)*. IEEE, 1–6. DOI: https://doi.org/10.1109/VTS.2018.8368646

[40] M. Rostami, F. Koushanfar, and R. Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* 102, 8 (Aug. 2014), 1283–1295. DOI: https://doi.org/10.1109/JPROC.2014.2335155

[41] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. 2018. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. Retrieved from https://openreview.net/forum?id=BkJ3ibb0.

[42] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM, 1528–1540. DOI: https://doi.org/10.1145/2976749.2978392

[43] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR'14)*. Retrieved from http://arxiv.org/abs/1312.6199.

[44] Aysa Fakheri Tabrizi, Nima Karimpour Darav, Shuchang Xu, Logan Rakai, Ismail Bustany, Andrew Kennings, and Laleh Behjat. 2018. A machine learning framework to identify detailed routing short violations from a placed netlist. In *Proceedings of the Design Automation Conference (DAC'18)*. ACM, New York, NY, 48:1–48:6. DOI: https://doi.org/10.1145/3195970.3195975

[45] Tianyu Gu, Kang Liu, Brendan Dolan-Gavitt, and Siddharth Garg. 2019. BadNets: Evaluating backdooring attacks on deep neural networks. *IEEE Access* 7 (2019), 47230–47244. DOI: https://doi.org/10.1109/ACCESS.2019.2909068

[46] J. Andres Torres. 2012. ICCAD'12 CAD contest in fuzzy pattern matching for physical verification and benchmark suite. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'12)*. 349–350.

[47] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick D. McDaniel. 2018. Ensemble adversarial training: Attacks and defenses. In *Proceedings of the 6th International Conference on Learning Representations (ICLR'18)*. Retrieved from https://openreview.net/forum?id=rkZvSe-RZ.

[48] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness may be at odds with accuracy. Retrieved from http://arxiv.org/abs/1805.12152.

[49] Wan-Yu Wen, Jin-Cheng Li, Sheng-Yuan Lin, Jing-Yi Chen, and Shih-Chieh Chang. 2014. A fuzzy-matching model with grid reduction for lithography hotspot detection. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* 33, 11 (Nov. 2014), 1671–1680. DOI: https://doi.org/10.1109/TCAD.2014.2351273

[50] Winston Haaswijk, Edo Collins, Benoit Seguin, Mathias Soeken, Frédéric Kaplan, Sabine Süsstrunk, and Giovanni De Micheli. 2018. Deep learning for logic optimization algorithms. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'18)*. 1–4. DOI: https://doi.org/10.1109/ISCAS.2018.8351885

[51] Zhiyao Xie, Yu-Hung Huang, Guan-Qi Fang, Haoxing Ren, Shao-Yun Fang, Yiran Chen, and Nvidia Corporation. 2018. RouteNet: Routability prediction for mixed-size designs using convolutional neural network. In *Proceedings of the International Conference on Computer-Aided Design (ICCAD'18)*. ACM, New York, NY, 80:1–80:8. DOI: https://doi.org/10.1145/3240765.3240843

[52] Weilin Xu, David Evans, and Yanjun Qi. 2018. Feature squeezing: Detecting adversarial examples in deep neural networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS'18)*. The Internet Society. http://wp.internetsociety.org/ndss/wp-content/uploads/sites/25/2018/02/ndss2018_03A-4_Xu_paper.pdf

[53] Xiaoqing Xu, Tetsuaki Matsunawa, Shigeki Nojima, Chikaaki Kodama, Toshiya Kotani, and David Z. Pan. 2016. A machine learning based framework for sub-resolution assist feature generation. In *Proceedings of the International Symposium on Physical Design*. ACM, 161–168.

[54] Haoyu Yang, Shuhe Li, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young. 2018. GAN-OPC: Mask optimization with lithography-guided generative adversarial nets. In *Proceedings of the Design Automation Conference (DAC'18)*. IEEE, 1–6. DOI: https://doi.org/10.1109/DAC.2018.8465816

[55] Haoyu Yang, Piyush Pathak, Frank Gennari, Ya-Chieh Lai, and Bei Yu. 2019. Hotspot detection using squish-net. In *Design-Process-Technology Co-optimization for Manufacturability XIII*, Vol. 10962. International Society for Optics and Photonics, 109620S. DOI: https://doi.org/10.1117/12.2515172

[56] Haoyu Yang, Jing Su, Yi Zou, Yuzhe Ma, Bei Yu, and Evangeline F. Y. Young. 2018. Layout hotspot detection with feature tensor generation and deep biased learning. *IEEE Trans. Comput.-Aided Design Integr. Circ. Syst.* (2018), 1–1. DOI: https://doi.org/10.1109/TCAD.2018.2837078

[57] Haoyu Yang, Jing Su, Yi Zou, Bei Yu, and Evangeline F. Y. Young. 2017. Layout hotspot detection with feature tensor generation and deep biased learning. In *Proceedings of the Design Automation Conference (DAC'17)*. ACM, 1–6. DOI: https://doi.org/10.1145/3061639.3062270

[58] Bo-Yi Yu, Yong Zhong, Shao-Yun Fang, and Hung-Fei Kuo. 2019. Deep learning-based framework for comprehensive mask optimization. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC'19)*. ACM, 311–316. DOI: https://doi.org/10.1145/3287624.3288749

[59] Cunxi Yu, Houping Xiao, and Giovanni De Micheli. 2018. Developing synthesis flows without human knowledge. In *Proceedings of the Design Automation Conference (DAC'18)*. ACM, 1–6. DOI: https://doi.org/10.1145/3195970.3196026

[60] Yen-Ting Yu, Ya-Chung Chan, Subarna Sinha, Iris Hui-Ru Jiang, and Charles Chiang. 2012. Accurate process-hotspot detection using critical design rule extraction. In *Proceedings of the Design Automation Conference (DAC'12)*. ACM, 1167–1172. DOI: https://doi.org/10.1145/2228360.2228576

[61] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th International Conference on Machine Learning*. 11.

Received June 2019; revised November 2019; accepted June 2020