

Attacking Split Manufacturing from a Deep Learning Perspective

Haocheng Li
CUHK

hcli@cse.cuhk.edu.hk

Satwik Patnaik
NYU

sp4012@nyu.edu

Abhrajit Sengupta
NYU

as9397@nyu.edu

Haoyu Yang
CUHK

hyyang@cse.cuhk.edu.hk

Johann Knechtel*
NYU Abu Dhabi

johann@nyu.edu

Bei Yu*
CUHK

byu@cse.cuhk.edu.hk

Evangeline F. Y. Young*
CUHK

fyyoung@cse.cuhk.edu.hk

Ozgur Sinanoglu*
NYU Abu Dhabi

ozgursin@nyu.edu

ABSTRACT

The notion of integrated circuit split manufacturing which delegates the front-end-of-line (FEOL) and back-end-of-line (BEOL) parts to different foundries, is to prevent overproduction, piracy of the intellectual property (IP), or targeted insertion of hardware Trojans by adversaries in the FEOL facility. In this work, we challenge the security promise of split manufacturing by formulating various layout-level placement and routing hints as vector- and image-based features. We construct a sophisticated deep neural network which can infer the missing BEOL connections with high accuracy. Compared with the publicly available network-flow attack [1], for the same set of ISCAS-85 benchmarks, we achieve 1.21× accuracy when splitting on M1 and 1.12× accuracy when splitting on M3 with less than 1% running time.

1 INTRODUCTION

Hardware becomes as vulnerable as software with the widespread globalization of design, synthesis, fabrication, and distribution of integrated circuits (ICs). Fabless IC companies have to rely on external, off-shore foundries for cost-effective access to advanced technology nodes, which enables numerous attack avenues on intellectual property (IP) including IP piracy [2], tampering with hardware monitors [3], and insertion of hardware Trojans [4].

The IARPA agency advocated split manufacturing to safeguard chip designs from potentially malicious foundries by splitting the designs into the front-end-of-line (FEOL) and back-end-of-line (BEOL) parts [5]. An untrusted, high-end foundry fabricates the device layer and a few lower metal layers, whereas a trusted, low-end facility, which is possibly in-house, integrates the BEOL on top of the FEOL by manufacturing the remaining metal layers [6]. However, merely splitting the designs into FEOL and BEOL may fall short in terms of security, as discussed next.

Traditional security-oblivious physical design tools place components close to each other in the FEOL layers when they are connected at the BEOL layers [7]. While delivering effective designs in terms of power, performance, and area (PPA), such an approach

leads to some information leakage so that, with an understanding of physical design tools, the structural information gathered from the FEOL layers can be utilized to infer the missing BEOL connections. This generic concept is known as a *proximity attack*. Rajendran *et al.* [8] demonstrated the first naïve proximity attack, where they leveraged the fact that interconnected modules are typically placed close by. The attack performed reasonably well for hierarchical designs with few nets between the modules but showed limited success for flat designs and large layouts. Wang *et al.* [1] proposed a network-flow model by setting the proximity as cost and capacitance as capacity. However, the attack incurs significant runtime and the network flow is relaxed to the naïve proximity attack when cell libraries has loose capacitance constraints. Zhang *et al.* [9] analyze the security of split manufacturing on industrial designs with random-forest classifier. However, their classifiers do not predict the BEOL connections directly, but generate a list of candidates with considerable size instead. For instance, when attacking layouts split on Metal 4 (M4), i.e., the FEOL contains M1–M4, their most successful classifier provides on average several hundreds or even thousands of candidates for each broken connection. Here it can become practically impossible to retrieve all correct connections.

Besides traditional classifiers, we believe that deep learning (DL) is a good match for attacking split manufacturing because attackers may have access to an extensive database of designs to train on. Among other applications, DL has been used to increase the effectiveness and efficiency in human-level control [10], object recognition [11], design-for-manufacturability (DFM) [12, 13], and routability prediction [14]. We caution that attacking split manufacturing with DL entails handling a large variety of data. Vector-based data are ranging, e.g., from signed gate displacements to unsigned wirelengths and from integral pin counts to floating point pin capacitances. Additionally, layout images can represent routing segments and their directions as well as congestions. Such image-based data naturally constitute rich information that can be useful for an advanced attack. One limitation of current neural network architectures is that they can only handle either vector- or image-based features, but not both types together. Accordingly, one challenge for our work is to combine both vector- and image-based features in a unified framework. Another limitation when applying DL for connection predicting is that traditional two-class classifiers can only predict the probability of each possible BEOL connection. However, in the physical-design reality, each sink is assigned to exactly one source. When selecting the source with the largest predicted connection probability, traditional classifiers

*Four authors contributed equally to this research.

can be easily misled by outlying predictions of negative samples. Furthermore, traditional classifiers also suffer from the imbalance between positive and negative samples.

In this paper, we leverage deep learning to thoroughly learn the characteristics of chip layouts arising from their physical design. To the best of our knowledge, this is the first deep-learning-based attack on split manufacturing that provides better results than the state-of-the-art non-learning-based attacks [1]. The major contributions of our work are as follows:

- We leverage deep learning (DL) for attacking split manufacturing. We design and train a sophisticated deep neural network architecture which can predict the missing BEOL connections for unknown FEOL layout with high accuracy.
- Our neural network makes use of vector-based and image-based layout features simultaneously.
- The proposed *softmax regression loss* allows our attack to directly and effectively select the best (i.e., most probable) BEOL connection among the relevant candidates.

The rest of the paper is organized as follows. Section 2 outlines the threat model and problem formulation. Section 3 contains our features for the DL attack. In Section 4, we describe the architecture and configuration of the proposed deep neural network whose effectiveness is verified in Section 5. Section 6 concludes the paper.

2 PRELIMINARIES

2.1 Threat Model

Consistent with prior work [1, 9], we assume that attackers have access to the full design information of the FEOL layers. Hence, they can identify the gates, FEOL routing, and the resulting but incomplete netlist. Also, they know the maximum load capacitances from the cell library and can estimate an upper bound for the delay. Further, consistent with prior work, we assume attacks take place while chips are being fabricated. Hence, oracle access is not available for the attacker. Although we acknowledge a recent work [15] where an oracle was leveraged to assist during the attack on split manufacturing, here we adopt a stronger threat model where the chip is either yet to be manufactured or is not available at all in the market. We also assume that an attacker has a database of layouts generated in a similar manner as the one under attack.

The objective of an attacker residing in the untrusted FEOL foundry is to decipher the missing BEOL interconnects solely from the available FEOL information. The corresponding goal is to reconstruct the design and ultimately to pirate the chip IP, overproduce the chip, or insert hardware Trojans.

2.2 Problem Formulation

Split layer is the top-most FEOL layer, while *virtual pins* are vias manufactured to connect the FEOL with the BEOL [9]. During split manufacturing, *fragments* are connected parts of FEOL wires, holding at least one virtual pin in the split layer. There are two different types of fragments as shown in Figure 1:

- *source fragment*: a driver/source along with outgoing fragments which are routed up until and within the split layer;
- *sink fragment*: an incoming fragment which is routed within the split layer and down towards the sink(s).

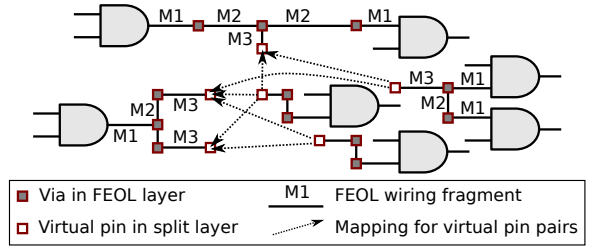


Figure 1: Terms for learning on split manufacturing layouts. Examples of virtual pin pairs are shown by dashed arrows.

For multi-fanout nets, the sinks may be routed together in the FEOL as one sink fragment or separately as several sink fragments.

Given a set of m sink fragments, each of which has c_1, c_2, \dots, c_m sinks, and a set of source fragments. *Virtual pin pairs* (VPPs) are mappings between sink fragment virtual pins and source fragments virtual pins. A VPP which is truly connected in the BEOL is called a *positive VPP*. Otherwise, it is called a *negative VPP*. The connection predicting problem is to select a VPP for each sink fragment maximizing the correct connection rate (CCR) [1], which can be thought of as the percentage of sink pins that are successfully restored. We define CCR as follows:

$$CCR = \frac{\sum_{i=1}^m c_i x_i}{\sum_{i=1}^m c_i}, \quad (1)$$

where $x_i = 1$ (0) when a positive (negative) VPP is selected for the i -th sink fragment.

3 FEATURE EXTRACTION

The BEOL is only available at training time, where the true connectivity is extracted to label VPPs as positive or negative ones. The FEOL is available for both the training and attacking phases. Hence, all features have to be tailored for the FEOL. We propose two feature categories for our DL attack: vector- and image-based features. We explain how to integrate these heterogeneous features into a unified neural network architecture in the next section.

3.1 Vector-based Features

3.1.1 Various Distances for VPPs. These features capture the working essence of physical-design tools, since gates to be connected are typically placed closer to each other. In accordance to routing principles, all distances are captured as Manhattan distances. For a VPP, the distances along the preferred and the non-preferred routing direction as well as their sum are all considered separately. Both signed and unsigned values are used to consider both the absolute and relative distances. To mitigate scaling issues across layouts, all distances are duplicated by encoding in the ratios of the chip width/height and half-perimeter, respectively.

3.1.2 Load Capacitance and Number of Sinks. These features track the total load value and the number of sinks for each VPP. As we are handling split or incomplete layouts, the load capacitances can only be defined by two bounds as follows:

- upper bound: maximum capacitance of the driver, as derived from the cell library (which is available to the attacker);

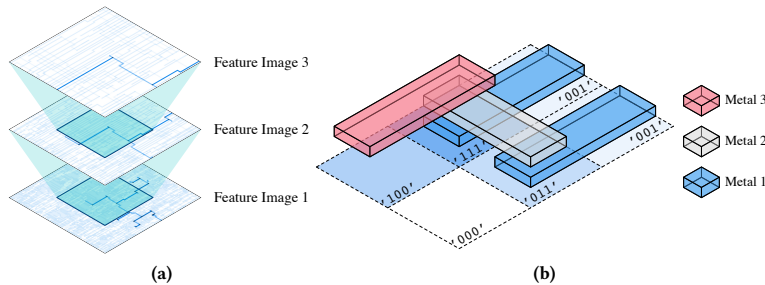


Figure 2: (a) Layout image scaling; (b) Layout image representation.

- lower bound: pin capacitance of the sinks connected within the sink fragment, plus wire capacitances of the two related source and sink fragments.

3.1.3 *FEOL Layer Wirelengths and Vias.* These features capture the wirelength contribution in each FEOL metal layer individually. Contributions are tracked separately for the two fragments of a VPP. Within each layer, all wire paths of a fragment are summed up. The number of vias in each FEOL cut layer is also considered.

3.1.4 *Driver Delay.* For each VPP, we track the driver delay based on the underlying timing paths. Note that timing paths obtained from split layouts can only provide lower bounds for delays, as the paths may be incomplete. Thus, this feature tends to become more meaningful for higher split layers, when more of the paths are already completed in the FEOL.

3.2 Image-based Features

For each virtual pin, we represent the routing layout of the local regions centering the virtual pin as gray-scale layout images. We consider three different scales with the same image shape but different precisions as shown in Figure 2(a). Each image is 99 pixels wide and high, representing 99×99 consecutive regions.

There are two properties of the routed wires which will be encoded in the layout images of a virtual pin: the nets they belong to and the layers they are routed on. Let m be the number of metal layers in the FEOL. The total number of bits in a pixel to represent the layout information is $2m$, and we call these bits *layer bits*. $2m$ layer bits are needed because wires of the same fragment as the virtual pin and wires from all other fragments are to be represented by different layer bits; the more significant m bits represent the routed wires of the virtual pin’s fragment and the less significant m bits represent the wires of other fragments. Wires of each type may exist in any routing layer of the FEOL, and wires on different layers are also represented by different layer bits. Since wires closer to the BEOL carry more information about the connection, those in higher metal layers are encoded in more significant bits while those in lower metal layers are encoded in less significant bits. Vias connecting two layers are represented in both layer bits.

More specifically, a 1 is assigned to the b -th bit with $b = m, \dots, 2m - 1$ in a pixel if the virtual pin’s fragment is routed in metal layer $b - m + 1$ in that region. Similarly, a 1 is assigned to the b -th bit with $b = 0, \dots, m - 1$ in a pixel if there is some wire or via arising from other fragments in metal layer $b + 1$ of that region. Figure 2(b) shows part of the image data which is representative for a part

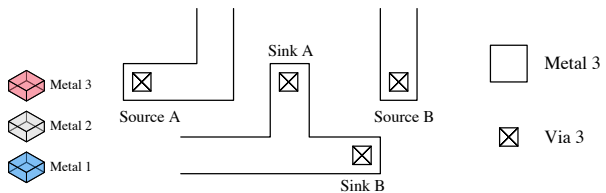


Figure 3: Direction example: Except VPP Source A–Sink B, all other VPPs are considered as candidates.

Table 1: VPP Preferences

Sk	Sc	Sk Prefers Sc	Sc Prefers Sk	Direction Criterion
A	A	✓	✗	✓
A	B	✓	✓	✓
B	A	✗	✗	✗
B	B	✓	✓	✓

of the layout as in Figure 1, where the split layer is M3, so that there may be wires on three different layers. Routed wires in the six consecutive regions bounded by the dashed lines are encoded into 2×3 pixels. Note that here we only show the values of the sixth, fifth and fourth layer bits at the corner of each region which together represent an exemplary virtual pin’s fragment.

4 LEARNING FRAMEWORK

In this section, we start by describing the VPP selection for data cleaning. We then demonstrate the neural network architecture and discuss our proposed softmax regression loss and its advantages.

4.1 Sample Selection

Due to an underlying tendency towards imbalanced datasets and long inference runtime, it is not practical to consider all possible VPPs because the correct connections are very few among all possible ones, which leads to a biased or inaccurate ML model. For N nets, even in the simplest scenario where (i) each FEOL wiring fragment holds only one virtual pin in the split layer and (ii) each net is split into exactly one source and one sink fragment, the sampling size is already N^2 , whereas only $\frac{1}{N}$ samples are true positives.

We select n candidate VPPs for each sink in training and testing based on three criteria. The first is the direction criterion. We apply a looser criterion than [1] to avoid neglecting positive VPPs, based on our observation that wires with non-preferred routing direction are relatively common in congested designs. For a VPP (p, q) , if q is on the opposite side of one of the wire segments directly connected to p , we then say the virtual pin p prefers virtual pin q . The preferences of the VPPs in Figure 3 are shown in Table 1. Note that preferences of virtual pins in the same fragment are evaluated separately. Now, our direction criterion is that a VPP is *not* considered as a candidate in case both source and sink pins *do not* prefer each other. The second criterion is non-duplication. If a sink fragment or source fragment have multiple virtual pins, for each pair of sink fragment and source fragment, only the VPP with the shortest distance apart

in the non-preferred routing direction of the split layer is considered as candidate. This is based on the domain knowledge that the net wirelengths are restricted to meet timing closure. The third criterion is distance. If the number of VPPs remaining is greater than n , the VPPs with shorter distance in the non-preferred routing direction of the split layer have higher priority to be selected. If multiple VPPs tie on the non-preferred routing direction distance, the distance in the preferred routing direction is considered as a tie-breaker.

4.2 Model Architecture

The input data for the model is a batch of features corresponding to a sink fragment including the vector-based features of n selected VPPs with the sink fragment, the image-based features of n source fragments in the related VPPs, and the image-based features of the sink fragment itself. The output data are scores for every VPPs in the batch. To handle vector- and image-based features in the same network, the proposed neural network illustrated in Figure 4 first extracts underlying features from heterogeneous input by processing vector-based features (shown in the upper left) and image-based features (shown in the upper middle) individually, and then processing them together (shown in the lower left) after concatenating the output of the vector and image part together.

For the image part of the network, note that the image-based features of the sink fragment are the same in the batch, so we only process them once to save runtime and its output is distributed to the output of every source images. Besides, all the image-based features go through the same shared network. The shared network contains twelve convolution layers (red colored, labeled as `conv`) and two fully connected layers (blue colored, labeled as `fc`). The output from the sink image is then concatenated with every output from the source images and the combination passes through one more 128-way fully connected layer. For the vector part of the network, vector-based features are first transformed by a 128-way fully connected layer. Then, there are four residual networks (ResNet) [16] blocks (purple colored, labeled as `res`). The output of a ResNet block is the sum of its input and the output of three fully connected layers as shown in the middle sub-figure of Figure 4. ResNet can resolve the gradient vanishing problem while training very deep neural networks. After that, the output from the image part is concatenated with the output from the vector-based features. There is one 128-way fully connected layer to down-size the combination. The network ends with three ResNet blocks and two more fully connected layers. The filter and parameter configuration of the neural network is listed in Table 2. Both fully connected layers and convolutional layers are followed by a leaky rectified linear unit (LReLU) $y = \max(0.01x, x)$ as activation, where x is the input and y is the output [17].

4.3 Softmax Regression Loss

Given a query of a batch of n VPPs with at most one positive VPP, the network predicts the connection probability s_1, s_2, \dots, s_n for each VPP. The connection predicting task is to determine the index of the correct VPP to be connected:

$$\arg \max_i s_i, \quad (2)$$

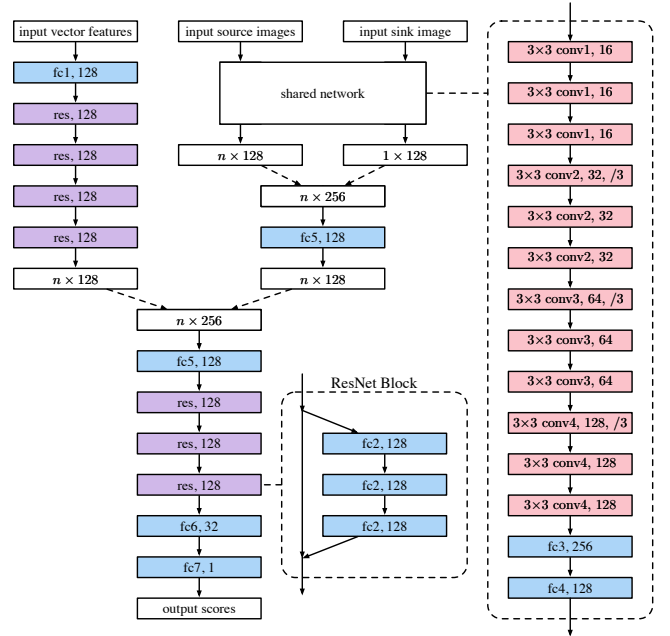


Figure 4: Neural Network Architecture.

Table 2: Neural Network Configuration.

Part	Layer	Parameter	Output
Vector part	fc1	27×128	$n \times 128$
	fc2	$[128 \times 128] \times 12$	$n \times 128$
Image part	conv1	$[3 \times 3, 16] \times 3$	$(n + 1) \times 99 \times 99 \times 16$
	conv2	$[3 \times 3, 32] \times 3$	$(n + 1) \times 33 \times 33 \times 32$
	conv3	$[3 \times 3, 64] \times 3$	$(n + 1) \times 11 \times 11 \times 64$
	conv4	$[3 \times 3, 128] \times 3$	$(n + 1) \times 4 \times 4 \times 128$
	fc3	128×256	$(n + 1) \times 256$
	fc4	256×128	$(n + 1) \times 128$
Merged part	fc5	256×128	$n \times 128$
	fc2	$[128 \times 128] \times 9$	$n \times 128$
	fc6	128×32	$n \times 32$
	fc7	32×1	$n \times 1$

as there can only be one source in a net. Simply modeling the VPP connection problem as a two-class classification is not appropriate. The main difference between our connection predicting problem and the classical regression problems is that we only care about the relative predicted probability between the only one positive VPP and the remaining negative ones, instead of their absolute values. Consequently, only the VPP with the largest predicted probability matters in the result. An outlying negative VPP prediction would easily mislead the matching.

Considering a two-class classification formulation, where the input of the neural network contains n VPPs with the same sink

fragment, the loss of the two-class classification is

$$l_r = -\frac{1}{n} \left(\log \frac{e^{s_i^+}}{e^{s_i^-} + e^{s_i^+}} + \sum_{j \neq t} \log \frac{e^{s_j^-}}{e^{s_j^-} + e^{s_j^+}} \right), \quad (3)$$

whose partial derivative with respect to each score of either class is

$$\frac{\partial l_r}{\partial s_j^+} = -\frac{\partial l_r}{\partial s_j^-} = \begin{cases} -\frac{e^{s_j^-}}{n(e^{s_j^-} + e^{s_j^+})} & \text{if } j = t, \\ \frac{e^{s_j^+}}{n(e^{s_j^-} + e^{s_j^+})} & \text{otherwise,} \end{cases} \quad (4)$$

where s_j^+ and s_j^- are the scores of connection and non-connection for the j -th source fragment with $1 \leq j \leq n$ and t is the index of the true connection. The partial derivative with respect to the i -th weight of either neuron in the last fully connected layer is

$$\frac{\partial l_r}{\partial w_i^+} = -\frac{\partial l_r}{\partial w_i^-} = \frac{1}{n} \left(\sum_{j=1}^n \frac{e^{s_j^+} x_{i,j}}{e^{s_j^-} + e^{s_j^+}} - x_{i,t} \right), \quad (5)$$

where $x_{i,j}$ is the i -th input value of the last fully connected layer for the j -th source fragment. Therefore, the score of each source fragment acts independently on the gradient. The coefficient of the positive part of the gradient, which is due to the negative samples, is limited to 1 so that the VPP with even the largest connection probability will not dominate the gradient. As a result, misprediction of one VPP, which significantly influences our desired output as in Equation (2), barely affects the average loss. Additionally, the numbers of positive and negative VPPs are imbalanced as most of the VPPs are negative samples. The negative part of the gradient, which is due to the only positive sample, is divided by the number of VPPs in the batch. Therefore, a two-class classification model has a serious imbalance problem as it can easily gain a high accuracy by simply classifying all VPPs as negative, which is meaningless.

To resolve these problems, we consider only one score s_j for the j -th source fragment with $1 \leq j \leq n$. We propose the following *softmax regression loss*

$$l_c = -\log \frac{e^{s_t}}{\sum_{j=1}^n e^{s_j}}, \quad (6)$$

whose partial derivative with respect to each score of connection is

$$\frac{\partial l_c}{\partial s_j} = \begin{cases} \frac{e^{s_j}}{\sum_{j=1}^n e^{s_j}} - 1 & \text{if } j = t, \\ \frac{e^{s_j}}{\sum_{j=1}^n e^{s_j}} & \text{otherwise.} \end{cases} \quad (7)$$

The partial derivative of our proposed loss with respect to the i -th weight of the only neuron in the last fully connected layer is

$$\frac{\partial l_c}{\partial w_i} = \frac{\sum_{j=1}^n e^{s_j} x_{i,j}}{\sum_{j=1}^n e^{s_j}} - x_{i,t}, \quad (8)$$

in which the two problems of the two-class classification model are resolved. Firstly, the source fragment with higher score contributes much more significantly in the gradient with an exponential factor. Let j_{max} be the index of the largest s_j . As the positive part of the loss is dominated by $x_{i,j_{max}}$, we have $\frac{\partial l_c}{\partial w_i} \approx x_{i,j_{max}} - x_{i,t}$. Secondly, the summation of the coefficients in the positive part equals to that of the negative part, so there is no imbalance issue. With these two advantages, the proposed softmax regression loss reflects better

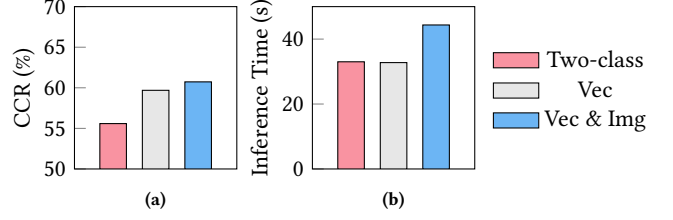


Figure 5: Comparison between different settings of techniques used; (a) Compare the average CCR; (b) Compare the average inference time.

our way of computing the output as in Equation (2), which is also supported by the empirical results.

5 EXPERIMENTAL RESULTS

We derive a total of 9 training and 5 validation designs from the ISCAS-85 [18], MCNC [19], and ITC-99 benchmarks [20]. We use the academic NanGate Open Cell Library [21] for all our experiments. Synopsys Design Compiler (DC) is used for synthesis. Cadence Innovus 16.15 performs the placement and routing. With regards to the to-be-attacked layouts, we use the same benchmarks as mentioned in [1] to ensure a fair comparison. All training, validation and to-be-attacked layouts are free from DRC violations. Once the layouts are generated, we export a Design Exchange Format (DEF) file which is then split after the first and third metal layer as two experimental sets to evaluate the performance with different split layer. We implement the feature extraction with C++ and train the model with Python and TensorFlow [22]. We select 31 VPPs for each sink fragment as the input of our DL attack based on the proposed criteria in Section 4.1. If the positive VPP is included in the collection, there are 30 negative VPPs and one positive VPP selected. If the positive VPP is not included, the predicted connection will definitely be wrong according to the criteria. We set each pixel in the image-based features representing a region of $0.05 \times 0.05 \mu\text{m}^2$, $0.1 \times 0.1 \mu\text{m}^2$, and $0.2 \times 0.2 \mu\text{m}^2$, respectively. The learning rate is set as 0.001 and decayed to 60% for every 20 epochs. We run all experiments on a 64-bit Linux machine with Intel Xeon 2.2GHz CPUs and an Nvidia Titan V GPU. We set the maximum runtime as 100,000 seconds (more than 24 hours) for all attacks and report the CCR (Equation (1)) as the accuracy metric.

In the first experiment, we illustrate the CCR for our proposed attacking method and the state-of-the-art method [1]. Table 3 lists the results. We evaluate the success of the network-flow attack ourselves using the binary released in [23] whose runtime exceeds the limit on several large designs. The CCR of our DL attack outperforms that of the state-of-the-art by 1.21× and 1.12× when splitting on M1 and M3, respectively. Our inference time (including feature extraction) is significantly shorter. Note that in the calculation of average, designs on which [1] times out are excluded for fairness.

In the second experiment, we verify the effectiveness of our proposed softmax regression loss and the image-based features. As shown in Figure 5, the baseline is splitting on M3 and using only the vector-based features with the loss function of the two-class classification. With the softmax regression loss, the average CCR

Table 3: Comparison with State-of-the-art Attack

Design	Metal 1						Metal 3					
	#Sk	#Sc	CCR (%)		Runtime (s)		#Sk	#Sc	CCR (%)		Runtime (s)	
			[1]	Ours	[1]	Ours			[1]	Ours	[1]	Ours
b11	738	296	9.05	10.03	1719.46	11.06	213	57	66.67	66.67	0.94	4.20
b13	430	215	10.42	17.91	130.82	7.53	88	52	42.05	70.45	0.44	3.55
b14	6338	2864	N/A	8.57	> 100000	77.62	2117	583	30.33	30.42	2576.42	16.08
b15_1	10176	3847	N/A	5.79	> 100000	130.30	4910	1235	26.42	24.24	38292.53	33.50
b17_1	32385	12479	N/A	4.08	> 100000	599.47	16190	4590	N/A	19.03	> 100000	157.61
b18	84292	33703	N/A	4.59	> 100000	2861.27	32719	9359	N/A	23.74	> 100000	453.66
b7	520	235	8.43	10.19	326.13	8.55	115	51	55.65	84.35	0.67	3.62
c1355	403	226	9.90	12.41	151.22	7.65	77	32	89.61	97.40	0.50	3.53
c1908	432	213	8.49	11.11	260.50	7.45	54	27	94.44	87.04	0.47	3.34
c2670	803	428	6.32	9.46	2251.82	11.70	206	120	54.85	58.74	1.48	4.64
c3540	1354	512	6.41	8.49	39187.25	17.55	452	124	54.87	51.11	7.39	5.42
c432	231	121	11.26	8.23	15.62	5.29	43	21	76.74	86.05	0.37	3.35
c5315	1919	847	7.50	9.33	94281.90	23.59	590	248	52.20	62.03	26.11	6.81
c6288	4124	2160	N/A	14.52	> 100000	49.64	551	78	63.16	61.52	7.13	4.22
c7552	2008	1108	12.10	11.11	48656.51	22.82	296	175	50.34	72.30	7.64	3.72
c880	460	234	11.09	13.91	568.99	6.31	77	37	71.43	76.62	0.74	2.34
Average Ratio			9.18	11.11	13889.37	10.67			59.20	66.35	2923.06	7.02
			1.00	1.21	1.000	0.001			1.00	1.12	1.000	0.002

is 1.07× that of the baseline. With additional image-based features, the average CCR further improves to 1.09×. The use of softmax regression loss improves the runtime marginally. The runtime when additionally using the image-based features is comparable to that of only using the vector-based features.

6 CONCLUSION

We presented an attack on split manufacturing using deep learning. Firstly, we demonstrated vector-based and image-based features and a neural network that can process these heterogeneous features simultaneously. We further proposed a softmax regression loss that directly reflects on the accuracy for the virtual pin pair matching problem of split manufacturing and, thereby, eliminates imbalance issues found in prior art. Compared with the state-of-the-art network-flow attack [1], the correct connection rate is improved by 1.21× and 1.12× when splitting on Metal 1 and Metal 3, respectively. Moreover, the runtime for our attack is significantly less, namely less than 1% of that of [1]. As a part of future work, we plan to extend our neural network to tackle industrial layouts which have been incorporated with various placement-based and/or routing-based defense strategies.

ACKNOWLEDGMENTS

This work is supported in part by The Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CUHK14202218) and Center for Cyber Security Abu Dhabi (CCS-AD) in New York University Abu Dhabi.

REFERENCES

[1] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran, “The cat and mouse in split manufacturing,” *IEEE TVLSI*, vol. 26, no. 5, pp. 805–817, 2018.
 [2] K. Shamsi, T. Meade, M. Li, D. Z. Pan, and Y. Jin, “On the approximation resiliency of logic locking and IC camouflaging schemes,” *IEEE TIFS*, vol. 14, no. 2, pp. 347–359, 2019.

[3] F. Durvaux and F.-X. Standaert, “From improved leakage detection to the detection of points of interests in leakage traces,” in *EUROCRYPT*, 2016, pp. 240–262.
 [4] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, “A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion,” *IEEE TCAD*, 2018.
 [5] C. McCants, “Trusted integrated chips (TIC),” Intelligence Advanced Research Projects Activity (IARPA), Tech. Rep., 2011.
 [6] Y. Bi, J. S. Yuan, and Y. Jin, “Beyond the interconnections: Split manufacturing in RF designs,” *Electronics*, vol. 4, no. 3, pp. 541–564, 2015.
 [7] H. Li, W.-K. Chow, G. Chen, E. F. Young, and B. Yu, “Routability-driven and fence-aware legalization for mixed-cell-height circuits,” in *Proc. DAC*, 2018, p. 150.
 [8] J. J. Rajendran, O. Sinanoglu, and R. Karri, “Is split manufacturing secure?” in *Proc. DATE*, 2013, pp. 1259–1264.
 [9] B. Zhang, J. C. Magaña, and A. Davoodi, “Analysis of security of split manufacturing using machine learning,” in *Proc. DAC*, 2018, pp. 141:1–141:6.
 [10] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
 [11] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in *Proc. ICCV*, 2017, pp. 2980–2988.
 [12] H. Yang, J. Su, Y. Zou, Y. Ma, B. Yu, and E. F. Young, “Layout hotspot detection with feature tensor generation and deep biased learning,” *IEEE TCAD*, 2018.
 [13] H. Yang, S. Li, Y. Ma, B. Yu, and E. F. Young, “GAN-OPC: Mask optimization with lithography-guided generative adversarial nets,” in *Proc. DAC*. IEEE, 2018, pp. 1–6.
 [14] Z. Xie *et al.*, “Routenet: Routability prediction for mixed-size designs using convolutional neural network,” in *Proc. ICCAD*, 2018, pp. 80:1–80:8.
 [15] S. Chen and R. Vemuri, “On the effectiveness of the satisfiability attack on split manufactured circuits,” in *Proc. VLSI-SOC*, 2018.
 [16] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. CVPR*, 2016, pp. 770–778.
 [17] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. ICML*, 2013.
 [18] M. C. Hansen, H. Yalcin, and J. P. Hayes, “Unveiling the ISCAS-85 benchmarks: A case study in reverse engineering,” *IEEE MDAT*, vol. 16, no. 3, pp. 72–80, 1999.
 [19] (1991) MCNC benchmarks. <http://vlsicad.cs.binghamton.edu/benchmarks.html>.
 [20] F. Corno, M. S. Reorda, and G. Squillero, “RT-level ITC’99 benchmarks and first ATPG results,” *IEEE MDAT*, 2000.
 [21] J. Knudsen, “Nangate 45nm open cell library,” *CDNLive, EMEA*, 2008.
 [22] M. Abadi *et al.*, “Tensorflow: a system for large-scale machine learning,” in *OSDI*, vol. 16, 2016.
 [23] (2017) Network flow attack. https://github.com/seth-tamu/network_flow_attack.