Estimating the Circuit De-obfuscation Runtime based on Graph Deep Learning

Zhiqian Chen*, Gaurav Kolhe†, Setareh Rafatirad§, Chang-Tien Lu*, Sai Manoj P D‡,
Houman Homayoun†, Liang Zhao§
* Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
† Department of Electrical and Computer Engineering, University of California, Davis, CA, USA
‡ Department of Electrical and Computer Engineering, George Mason University, VA, USA
§Department of Information Science and Technology, George Mason University, VA, USA
*{czq, ctlu}@vt.edu †{gskolhe, hhomayoun}@ucdavis.edu {‡spudukot, §srafatir, §lzhao9}@gmu.edu

***Abstract*—Circuit obfuscation has been proposed to protect digital integrated circuits (ICs) from different security threats such as reverse engineering by introducing ambiguity in the circuit, i.e., the addition of the logic gates whose functionality cannot be determined easily by the attacker. In order to conquer such defenses, techniques such as Boolean satisfiability-checking (SAT)-based attacks were introduced. SAT-attack can potentially decrypt the obfuscated circuits. However, the deobfuscation runtime could have a large span ranging from few milliseconds to a few years or more, depending on the number and location of obfuscated gates, the topology of the obfuscated circuit and obfuscation technique used. To ensure the security of the deployed obfuscation mechanism, it is essential to accurately pre-estimate the deobfuscation time. Thereby one can optimize the deployed defense in order to maximize the deobfuscation runtime.**

However, estimating the deobfuscation runtime is a challenging task due to 1) the complexity and heterogeneity of the graph-structured circuit, 2) the unknown and sophisticated mechanisms of the attackers for deobfuscation, 3) efficiency and scalability requirement in practice. To address the challenges mentioned above, this work proposes the first machine-learning framework that predicts the deobfuscation runtime based on graph deep learning. Specifically, we design a new model, ICNet with new input and convolution layers to characterize the circuit's topology, which is then integrated by composite deep fully-connected layers to obtain the deobfuscation runtime. The proposed ICNet is an end-to-end framework that can automatically extract the determinant features required for deobfuscation runtime prediction. Extensive experiments on standard benchmarks demonstrate its effectiveness and efficiency beyond many competitive baselines.

I. Introduction

The considerable high capital and operational costs on semiconductor fabrication have motivated most semiconductor companies to outsource it is a fabrication to off-shore foundries. Despite the reduced cost and other benefits, this trend has led to ever-increasing security risks such as IC counterfeiting, piracy and unauthorized overproduction by the contract foundries [23]. The overall financial risk caused by such counterfeit and unauthorized ICs was estimated to be over $169 billion per year [16]. The major threats from the attackers arise from reverse engineering (RE) an IC and fully identifying its functionality through brute force approaches by applying test inputs and obtaining outputs. To prevent such reverse engineering, *Hardware obfuscation* techniques have been extensively researched in recent years [30]. The general idea is to introduce the ambiguity in the functionality of the IC through obfuscation so that the while preserving the original functionality. Such techniques were highly effective until the advent of advanced attacking techniques. This is based on the fact that there are limited types of gates (e.g., AND, OR, XOR) in IC, so the attackers can just brute force all the possible combinations of types for all obfuscated gates to find out the one that functions identically to the targeted IC to be deobfuscated. As brute force is usually prohibitively expensive, more recently, efficient methods such as Boolean satisfiability problem (SAT)-based attacks have been proposed, which have attracted enormous attention [14].

The runtime of the SAT attack to reverse engineer the IC highly depends on the complexity of the obfuscated IC, which can vary from milliseconds to years or more depending on the number and location of obfuscated gates. Therefore, a successful obfuscation defence is to increase the amount of time (i.e., many years) required to reverse engineer the design. However, obfuscation comes at a substantial cost in finance, power, and space, and such trade-off requires us to search for optimal positions instead of purely increasing their quantity. The obfuscation policy tries to select a set of gates such that maximum obfuscation can be achieved while incurring minimal overheads. Although such selection can significantly influence the deobfuscation runtime, however, until now it is still generally based on human heuristics or experience, which is seriously arbitrary and sub-optimal [7]. This is major because it is unable to "try and error" all the different ways of obfuscation, as there are millions of combinations to try and the runtime for each execution (i.e., to run the attacker) can be days, weeks, or years.

To address this issue, this paper focuses on efficient and scalable ways to estimate the runtime of an attacker to reverse engineer an obfuscated IC. This research topic is highly under-explored because of its significant challenges: **1) Difficulty in characterizing the hidden and sophisticated algorithmic mechanism of attackers.** Over the recent years, a large number of deobfuscation methods have been proposed with various techniques [7]. In order to practically defeat the obfuscation schemes, methods with more and more sophisticated theories, rules, and heuristics have been proposed and adopted. The behaviour of such highly nonlinear and strongly-coupling systems is prohibitive for conventional simple models (e.g., linear regression and support vector machine [1]) to characterize. **2) Challenge in extracting determinant features from discrete and graph-structured IC.** The inputs of the runtime estimation problem is the IC and the selected gates for obfuscation, where the first input is a heterogeneous graph while the second is a vector with discrete values. Conventional feature extraction methods are not intuitive to be applied to such type of data without significant information loss. Hence, it is highly challenging to instantly formulate and seamlessly integrate them as mathematical forms that can be input to conventional computational and machine learning models. **3) Requirement on high efficiency and scalability for deobfuscation runtime estimation.** The key to the defence against deobfuscation is the

978-3-9819263-4-7/DATE20/©2020 EDAA

speed. The faster the defender can estimate the deobfuscation runtime for each candidate set of obfuscated gates, the more candidate sets the defender can evaluate, and hence the better the obfuscation effect will be. Moreover, the estimation speed of deobfuscation runtime must not be sensitive to different obfuscation strategies in order to make the defender strategy controllable.

This work addresses all the above challenges and proposes the first generic framework for deobfuscation runtime prediction, based on graph deep learning techniques. In the recent years, deep learning methods in complex cognitive tasks such as object recognition and machine translation have achieved immense success [27], [12], which motivates the generalization of it into graph-structured data [8]. By concretely formulating ICs and the obfuscated gates as multi-attributed graphs, this work innovatively leverages and extends the state-of-the-art graph deep learning methods such as Graph Convolutional Neural Networks (GCN) [8] to instantiate a graph regressor. Such end-to-end deep graph regressor can characterize the underlying and sophisticated cognitive process of the attacker for deobfuscating the ICs. To adopt the powerfulness of GCN and handle the aforementioned issues, we extend it by adjusting the connectivity representation inspired by domain facts. Our enhanced GCN can automatically extract the discriminative features that are determinants to the estimation of the deobfuscation runtime to achieve accurate runtime prediction. After being trained, the prediction based on this deobfuscation runtime estimator just runs instantly fast by simply performing a feed-forward propagation process. The major contributions of this paper are:

- Proposing a new framework, ICNet, for deobfuscation runtime estimation based on graph deep learning.
- Developing a new multi-attributed graph convolutional neural network for graph regression.
- Conducting systematical experimental evaluations and analyses on real-world datasets (ISCAS-85 benchmark).

The rest of the paper is organized as follows. Section II reviews the existing work. Section III elaborates proposed graph learning model for SAT runtime prediction. In Section IV, experiments on real-world data are presented. This paper concludes by summarizing the study's important findings in Section V.

II. Background and Related Work

We discuss the logic obfuscation and SAT attacks followed by graph convolutional networks and the relevant works.

A. Logic Obfuscation and SAT Attacks

Logic obfuscation often referred to as logic locking [29] is a hardware security solution that facilitates to hide the IP using key-programmable logic gates The activation of the obfuscated IP is accomplished in a trusted regime before releasing the product into the market, thereby reducing the probability to obtain the secret configuration keys by the attacker. During the activation phase, the correct key is applied to these key-programmable gates to recover the correct functionality of the IC/IP. Besides, the correct key will be stored in the IC in a tamper-proof memory. Although obfuscation schemes try to minimize the probability of determining the correct key by an attacker, thereby curbing the ongoing piracy of the legitimate IPs. However, SAT attack shows that the contemporary obfuscation schemes can be broken [24] to retrieve the correct key. In order to perform SAT attack, the attacker is required to have The SAT attack first tries to find the Distinguishing Input Patterns (DIP) X_i, which when applied as the input can produce different outputs (Y_i) such that ($Y_1 \neq Y_2$) when different key values are applied (K_1, K_2). This DIP can then be used to distinguish the correct and incorrect keys. The number of DIPs discovered during the SAT-based attack is the same as the number of iterations needed to unlock the obfuscated design. In each iteration, a constraint is added to SAT solver, until SAT solver cannot find a satisfying assignment. This results in finding the correct key.

Different SAT-hard schemes such as [9], [10] are proposed Furthermore, new obfuscation schemes that focus on non-Boolean Behaviour of circuits [28], that are not convertible to an SAT circuit is proposed for SAT resilience. Some of such defences include adding cycles into the design [18]. By adding cycles into the design may cause that the SAT attack gets stuck in the infinite loop, however, advanced SAT-based attacks such as cycSAT [32] can extract the correct key despite employing such defences. To ensure that the proposed defence ensures robustness against SAT attacks, the defenders need to run the rigorous simulations which could range from few minutes up to a few days. Furthermore, this can be exacerbated when the defender verifies for large and real-world circuits. This work proposes the use of graph convolutional networks (GCNs) to alleviate the need to run the attack to verify whether the defence is strong enough or not. The work in [19] utilizes neural network with single-bit supervision to predict whether a given circuit in Conjunctive Normal Form (CNF) can be decrypted or not. However, this is limited to determining for few kinds of SAT-solvers, but cannot be applied to SAT-hard solutions such as SMT-SAT [31], a superset of SAT attacks. However, with the proposed graph convolutional network (GCN) based predictor, the defender can determine the deobfuscation time in a single run of GCN, which consumes a few seconds. We introduce the GCN below.

B. Graph Convolutional Networks

Spectral graph theory is the study of the properties of a graph in relationship to the characteristic polynomial, eigenvalues, and eigenvectors of matrices associated with the graph. Many graphs and geometric convolution methods have been proposed recently. The spectral convolution methods [4], [8] are the mainstream algorithms developed as the graph convolution methods. Their theory is based on the graph Fourier analysis [20]. The polynomial approximation is firstly proposed by [6]. Inspired by this, graph convolutional neural networks (GCNs) ([4]) is a successful attempt at generalizing the powerful convolutional neural networks (CNNs) in dealing with Euclidean data to modelling graph-structured data. Kipf and Welling proposed a simplified type of GCNs [8], called graph convolutional networks (GCNs). The GCN model naturally integrates the connectivity patterns and feature attributes of graph-structured data and outperforms many state-of-the-art methods significantly.

Therefore, it is promising to apply GNNs for the circuit problem, since ICs can be naturally represented using a graph with connectivity among gates.

III. Proposed Model for Runtime Prediction

This section introduces the problem setting, and we present the deobfuscation time prediction through the proposed ICNet.

A. Problem Setting

First, a circuit is modeled as a graph network: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a set of n vertexes (gates), $\mathcal{E}$ represents links among gates and $\mathcal{W} = [w_{ij}] \in \{0,1\}^{n \times n}$ is an unweighted adjacency matrix. A signal $\mathbf{X}$ defined on the nodes is regarded as a vector $\mathbf{X} \in \mathbb{R}^{n \times F}$. A graph structure representation, i.e., combinatorial graph Laplacian, is defined as $\mathbf{L} = D - \mathcal{W} \in \mathbb{R}^{n\times n}$ where D is degree matrix. Accordingly, we formulate the estimation of running time on IC as a regression task. Specifically, the model accepts graph structure along with gate features as input, and predict the running time:

$$Y = f(\mathcal{G}, \mathbf{X})\Theta, \tag{1}$$

where f is a function integrating graph structure $\mathcal{G}$ and gate feature $\mathbf{X}$. $\mathcal{G}$ is often represented by graph Laplacian $\mathbf{L}$ in graph theory [2] Θ indicates the parameters of fully neural network layers connecting the actual runtime Y and f. The purpose of Θ is (1) fitting dimension with Y and (2) generalizing the logic pattern between Y and f. The goal of 1 is to learn f and Θ so that the difference between Y and $f(\mathcal{G}, \mathbf{X})\Theta$ is minimized. However, there exists no straightforward relationship among the number of obfuscated gates, type of obfuscated gates and other factors to determine the deobfuscation time. This makes it harder to efficiently estimate the SAT-attack runtime with traditional machine learning models. Hence, we survey a thread of works called graph neural networks or geometric deep learning to address the problem of deobfuscation estimation, as the netlist can be perceived as the graph representation of various logical elements.

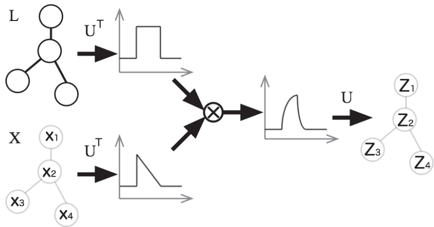


Figure 1: GCN workflow.

Graph convolutional network (GCN) is a recently emerging technique that integrate graph structure and node attributes, and its general process is performed as follows: it determines a vertex complete set of orthonormal Eigen vectors (frequency components) and their associated ordered real non-negative eigenvalues identified as the weights of these frequencies components. Specifically, the Laplacian is first diagonalized by the Fourier basis $\mathbf{U}^\intercal$: $\mathbf{L} = \mathbf{U}\,\Lambda\,\mathbf{U}^\intercal$ where Λ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues, i.e., $\Lambda_{ii} = \lambda_i$. The graph Fourier transform of a signal $\mathbf{X} \in \mathbb{R}^{n\times F}$ is defined as $\hat{\mathbf{X}} = \mathbf{U}^\intercal\,\mathbf{X} \in \mathbb{R}^n$ and its inverse as $\mathbf{X} = \mathbf{U}\,\hat{\mathbf{X}}$ [20], [21]. To enable the formulation of fundamental operations such as filtering in the vertex domain, the convolution operator on graph is defined in the Fourier domain such that $f_1 * f_2 = \mathbf{U}\,[(\mathbf{U}^\intercal f_1) \otimes (\mathbf{U}^\intercal f_2)]$, where $\otimes$ is the element-wise product, and f_1/f_2 are two signals defined on vertex domain. The intuitive workflow of GCN is shown in Figure 1. It follows that a vertex signal $f_2 = \mathbf{X}$ (gate features) is filtered by spectral signal $\hat{f}_1 = \mathbf{U}^\intercal f_1 = \mathbf{g}$ (graph structure) as:

$$\mathbf{g} * \mathbf{X} = \mathbf{U}\,[\mathbf{g}(\mathbf{\Lambda}) \odot (\mathbf{U}^\intercal f_2)] = \mathbf{U}\,\mathbf{g}(\mathbf{\Lambda})\,\mathbf{U}^\intercal\,\mathbf{X}\,.$$

B. Proposed Model: ICNet

Our proposed method, namely ICNet, is a neural network that is based on graph convolution operator. As shown in Figure 2, ICNet encodes the obfuscated circuit into two components:

- **Graph Structure** $\mathcal{G}$: Complete set of local connections are often used to represent the graph structure [2]. Typically, a graph Laplacian is employed, in this work since it contains gate-wise connections.
- **gate features X**: Gate-level information is encoded as numerical vector as input feature. Such information could include gate type, whether it is obfuscated and so on.

By applying the GCN, we can easily build a model to learn the relationship between the circuit and deobfuscation time automatically. However, GCN suffers from several issues: **(1)** the original graph convolutional operator is not suitable for the circuit since the graph Laplacian will make the graph convolutional operator behaviour as label propagation, i.e., the attributes of each gate are similar to its neighbours. This is called the smoothness assumption [13], and it does not fit the fact that gate type or encryption location of each gate does not determine its neighbours' related attributes in theory. This issue is due to that graph Laplacian matrix is used during graph convolution operation, which counts each node as -N_i (i is the index of the row in graph Laplacian), and counts the weighted sum of its neighbours as N_i. Consequently, they are cancelled out when gate representation are aggregated using sum, and the model can hardly learn the relationship between their sum (residues) and actual runtime. **(2)** The default setting of GCN aggregate gates and their features using the mean function, which is not supported by any domain knowledge, and not likely to cover the actual pattern on features or gates. To solve these issues, our model employs several policies to enhance the traditional GCN for circuit learning.

- **Graph Representation** $\mathcal{G} = A$: Our model uses adjacency matrix A instead of graph Laplacian. This representation can avoid intrinsic smoothness assumption which is not compatible with ICs.
- **Feature Aggregation**(Θ_{feat}): The mean function is a typical methods for aggregating node feature. However, the mean function does not consider the quantity of sum. A more flexible way is to learn feature aggregation by a neural network automatically.
- **Gate Aggregation**(Θ_{gate}): similarly, mean function can also be used to aggregate gate representation. Due to the complicated real-world aggregation, another neural network is designed to learn the gate aggregation function for more flexibility.

Our model is based on GCN which simplify the layer parameters of graph convolutional operator and applies an approximate technique to boost the efficiency. GCNs, as a state-of-the-art deep learning method for the graph, focus on processing graph signals defined on undirected graphs. According to the analysis above, graph Laplacian is replaced with adjacency matrix. To fit whole-graph level regression task, the proposed method designs two aggregation neural networks. Formally, it is denoted as:

$$\begin{aligned}
Y &= f(\mathcal{G}, \mathbf{X})\Theta && \text{(GNN definition Eq. 1)}\\
&= \underbrace{\mathbf{GCN}(\mathcal{W}, \mathbf{X})}_{\text{f=GCN}}\,\Theta && \text{(apply GCN with adjacency matrix)}\\
&= \mathbf{GCN}(\mathcal{W}, \mathbf{X})\Theta_{feat}\Theta_{gate} && (\Theta \rightarrow \{\Theta_{feat}, \Theta_{gate}\})\\
&= \underbrace{\sigma(\mathcal{W}\,\mathbf{X}\,\Theta_{\mathbf{GCN}})}_{\text{GCN}}\Theta_{feat}\Theta_{gate}, && \text{(rewrite GCN in matrix form)}
\end{aligned} \tag{2}$$

where activation σ is implemented by ReLU function. The running time tends to grow at an exponential rate as the number of encrypted gates increase. Therefore, the model is modified as:

$$Y = \exp(\mathcal{W}\,\mathbf{X}\,\Theta_{\mathbf{GCN}}\Theta_{feat}\Theta_{gate}) \tag{3}$$

As illustrated in Fig. 2, the proposed ICNet conducts two graph convolutional operations (GCN) to fuse the information

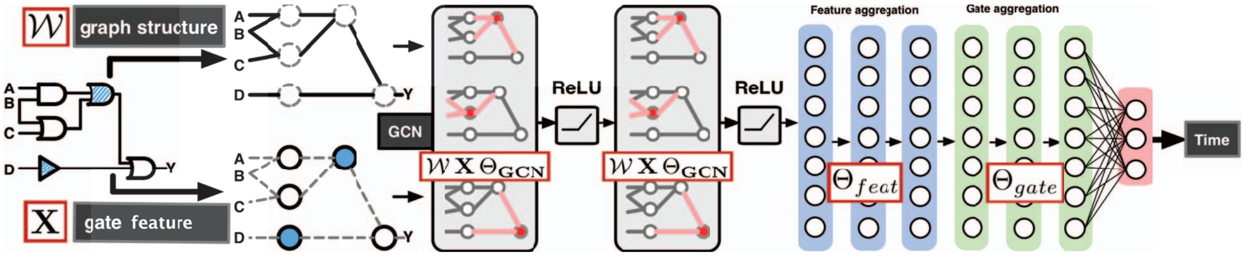


Figure 2: Illustration of ICNet structure: Two graph convolutions ($\mathcal{W}\mathbf{X}\Theta_{\mathbf{GCN}}$) followed by ReLU activation, and attention layers for features (Θ_{feat}) and gate (Θ_{gate}) respectively.

from graph structure and gate features. Then two sets of neural networks are performed for the feature and gate aggregation. To further increase the model's interpretability, we replace fully connected layers Θ_{feat} and Θ_{gate} as follows: Generally, *sum* or *mean* function is a typical method for aggregating node attribute into lower dimensional vector. This treats voting from each gate equally, which is not fit in theory. For example, the encrypted gate should be weighed higher, since it impose more difficulty on obfuscation task; gate types also have a significant impact on runtime [23]. Therefore, a more flexible way is to build a neural network to automatically learn attribute aggregation. To fit the whole-graph level regression task, the proposed method designs two aggregation neural components based on soft attention mechanism [26] for feature-level and gate-level. Formally, the feature based attention is calculated as:

$$a_i = \frac{exp(e_i)}{\sum_i exp(e_i)}, e_i = \sum_i \theta_i \mathbf{F}_i, \tag{4}$$

where $\mathbf{F}_i$ represents ith feature after $\mathbf{GCN}$, θ_i is the weight parameter for $\mathbf{F}_i$, a_i is the corresponding attention and thereby the output of this layer is $\sum_i a_i \mathbf{F}_i$. This attention shows which feature contributes more to the obfuscation time. Similarly, gate-wise attention is utilized for gate-level aggregation by setting $\mathbf{F}_i$ to ith gate in (4).

Algorithm 1: ICNet

```
Input: An integrated circuit graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, gate features
       set: $x_j(i)$, $i \in 1, 2, ..., |\mathcal{V}|$ for each encryption instance
       $D_j$, the real runtime $Y_j$ for instance $D_j$
Output: A neural network function with parameters $\Theta_{\mathbf{GCN}}$,
        $\Theta_{feat}$ and $\Theta_{gate}$
1  // Data preparing
2  Calculate $\mathcal{W}$ which is the adjacency matrix of $\mathcal{G}$
3  Split encryption instances $D$ into training set $D_{train}$ and
     testing set $D_{test}$
4  Split both $D_{train}$ and testing set $D_{test}$ into batch set $d_{train}$
     and testing set $d_{test}$
5  // Update ICNet
6  $\theta = \{\Theta_{\mathbf{GCN}}, \Theta_{feat}, \Theta_{gate}\}$
7  Initialize $\theta$ with Gaussian or uniform distribution.
8  repeat
9      Randomly select one $d_{train} = x_{d1}, x_{d2}, ...$
10     Calculate predicted runtime $\hat{Y}$                ▷ Eq. 3 and 4
11     Calculate residues $\delta = Y - \hat{Y}$
12     Compute derivatives to update parameters: $\theta \leftarrow \theta + \beta \nabla_\theta \delta$,
         where $\beta$ is learning rate
13 until $\delta$ convergence;
```

C. Algorithm description

The Algorithm 1 first prepare graph adjacency as circuit connection representation (line 2). To fit the machine learning schema, the whole dataset is split into training and testing dataset. Each dataset is then split into small batch size to improve learning efficiency (line 3-4). ICNet training is an iterative process which updates the model until the residues are small enough or converged (line 6-13). First, the model parameters are initialized by Gaussian or uniform distribution. In each iteration, a batch of the training set is selected randomly. By equation 3, the model computes the predicted runtime (line 10) and then calculates the residues between real runtime and prediction (line 11). Following normal deep learning schema, the model update parameters by the derivatives regarding the parameters themselves with learning rate (line 12).

IV. Evaluation

This section elaborates evaluation of the proposed method ICNet with competitive baselines including: Graph deep learning methods: GCN [8], ChebNet [4]. The input of these models above is exactly same as our model. We also compare against several state-of-the-art regression models[1]: Linear Regression (LR), LASSO [25], Epsilon-Support Vector Regression(SVR) (Two kernels were applied: polynomial (P) and RBF (R)), [22], Ridge Regression (RR) [17], Elastic Net (EN) [33], Orthogonal Matching Pursuit (OMP) [15], SGD Regression, Least Angle Regression (LARS) [5], Theil-Sen Estimators (Theil) [3]. These regression models does not model graph using Laplacian or adjacency matrix, since they can only accept feature vector. Therefore, the input are encoded as mean or sum on concatenation of Laplacian or adjacency matrix and gate features.

A. Data processing

The datasets are obtained by running SAT algorithm [24], [23] on real-world ISCAS-85 benchmark: First, we take a circuit and select a random gate and replace it with LUT of fixed size (LUT size 4 in current work). To deobfuscate, we implement SAT attack algorithm [24], [23] with the obfuscated circuit netlist as input. We monitor the time that SAT-attack takes to decode the key, which is the deobfuscation time. The proposed model is evaluated on two datasets: **Dataset 1**: the total number of the encryption location ranges from 1 to 350, this is for testing if the model is sensitive to the number of encrypted quantity of gates. **Dataset 2**: the total number of the encryption location ranges from 1 to 3, this is for testing if the model can handle very small value.

The circuit in the experiments is the same, and the total gate number of the circuit is 1529. For graph deep learning methods, the graph is represented using Laplacian matrix or adjacency matrix, while for general regression baselines, the graph Laplacian or adjacency matrix is summed or averaged across gates. Though the evaluations showed here are mere proof-of-concept of how powerful the proposed GCN based deobfuscation runtime prediction is, it can be applied to an SAT-hardening solution utilizing any replacement policy, LUT size and other SAT parameters, by retraining GCN.

[1] https://scikit-learn.org/stable/modules/linear_model.html

B. Experiment configuration

The features of gate used in experiments include: **gate mask**: if the gate is encrypted, the value is set to 1, otherwise 0. and **gate type**: the gate type include {AND, NOR, NOT, NAND, OR, XOR}, they are encoded using one-hot coding, such as [1,0,0,0,0,0] for AND and [0,1,0,0,0,0] for NOR gate.

For graph deep learning model (ChebNet and ICNet), the graph structure is represented using graph Laplacian matrix or adjacency matrix. These model employ ADAM [11] optimizer and will stop learning when the learning loss is converged. The implementation of our model will be available online. All the baselines and the proposed model are tested on two different feature set, since gate type is useful or not is unknown.: **Location**: Only the gate mask is included. **All features**: Besides gate mask, gate type is also included.

For node aggregation, we apply sum, and $mean$ since they are popular. Deep learning model can have another node aggregation method, i.e., learning by a neural network automatically. Therefore, in the results, ChebNet-NN and ICNet-NN denote the automatic version. It is expected that a deep neural network can learn an optimal aggregation which is not worse than our assumption, i.e., sum or mean.

Table I: Regression Performance (MSE) on Dataset 1

	Location		All feat	
Method	Sum	Mean	Sum	Mean
SVR RBF	1.6791	0.6784	1.6675	0.6739
SVR Poly	0.1913	2.1890	0.1696	2.2091
SGD	2.1450e+25	2.1823	1.0430e+26	2.2072
LR	0.2839	0.2284	0.2449	0.2253
RR	0.2309	2.1508	0.2058	2.1738
LASSO	0.9213	2.1843	1.0127	2.2083
EN	0.5763	2.1843	0.6409	2.2083
OMP	1.8182	1.9192	1.8651	2.0337
LARS	1.9968	2.1277	2.0434	2.1833
Theil	0.2948	0.2238	0.2385	0.2277
ChebNet	0.1484	8.8370e+33	0.1761	0.1760
ChebNet-NN	0.17858		3.8549e+27	
GCN	0.3364	0.4149	0.2496	0.3290
GCN-NN	0.1811		0.1606	
ICNet	0.1534	0.1256	0.2390	0.1902
ICNet-NN	**0.0843**		**0.1367**	

C. Regression Results

In the dataset 1 experiment (Table I, all methods achieved acceptable mean square error (MSE) except SGD (sum) which did not learn a reasonable model to predict the runtime, since the value is tremendous (at e+25/+26 scale). Most regression methods are sensitive to the aggregation method. For example, only using location feature, MSE of RR is 0.2309 when using sum, but it got 2.1508 when using the mean function. Sensitive models include SVR, LASSO, and EN. The best of the regression baselines is SVR (ploy), which achieved MSE of 0.1913. On the other hand, ChebNet is slightly better than the best regression model. However, ChebNet is not stable and sensitive to the aggregation method and feature set since it may yield a substantial error. Our proposed ICNet-NN is stable to the feature and aggregation setting and outperformed all the other methods, i.e., 0.0843 of MSE. Note that ICNet-NN is better than ICNet with sum or mean function, which demonstrates that there exists a better aggregation method, and graph neural network can learn it automatically. ICNet is always better than GCN under any settings, which shows that our improvement based on GCN works on circuit scenario. While in the dataset 2 (Table II), it is more challenging, since all the runtime are small and the model has to be very precise to achieve low MSE. All methods at almost the same level of MSE. Once again, some of the regression models are not stable such as SGD and LR. Graph deep learning method includes ChebNet and ICNet still at the best error level. ChebNet can achieve the best level but sensitive to the settings (i.e., location or all feature), while ICNet is insensitive to this setting. Therefore, ICNet is more stable than GCN and ChebNet, since the difference of MSE between location and all feature is smaller than that of ChebNet or GCN. Under all feature setting, ICNet-NN is still the best method, and it outperformed its mean and sum version.

Table II: Regression Performance (MSE) on Dataset 2

	Location		All feat	
Method	Sum	Mean	Sum	Mean
SVR RBF	0.0051	0.0048	0.0050	0.0051
SVR Poly	0.0048	0.0048	0.0048	0.0051
SGD	7.6301e+25	0.0045	2.0675e+26	0.0049
LR	6.9063e+23	4.6521e+20	7.2916e+25	5.8600e+23
RR	0.0070	0.0045	0.0065	0.0049
LASSO	0.0047	0.0045	0.0046	0.0049
EN	0.0047	0.0045	0.0046	0.0049
OMP	0.0047	0.0045	0.0045	0.0049
PAR	0.0054	0.1918	0.0051	0.3143
LARS	0.0047	0.0045	0.0046	0.0049
Theil	N/A	N/A	N/A	N/A
ChebNet	0.0047	0.0045	4.3570e+28	0.0048
ChebNet-NN	**0.0043**		0.0047	
GCN	0.0061	0.0046	0.0048	0.0050
GCN-NN	0.0050		0.1606	
ICNet	0.0049	0.0047	**0.0040**	0.0043
ICNet-NN	0.0051		0.0048	

The proposed method, ICNet, not only predicted the value very precisely but also with small variance. The runtime of ICNet is 1.1336 seconds on average, ranging from 1 to 2 seconds on dataset 1 and 2. This is because runtime of ICNet only depends on its parameter number. The instance with the largest runtime on dataset 1 and 2 spends 2411.11 seconds by actual solver. Therefore, ICNet can save 99.95% of solver's time to get accurate runtime.

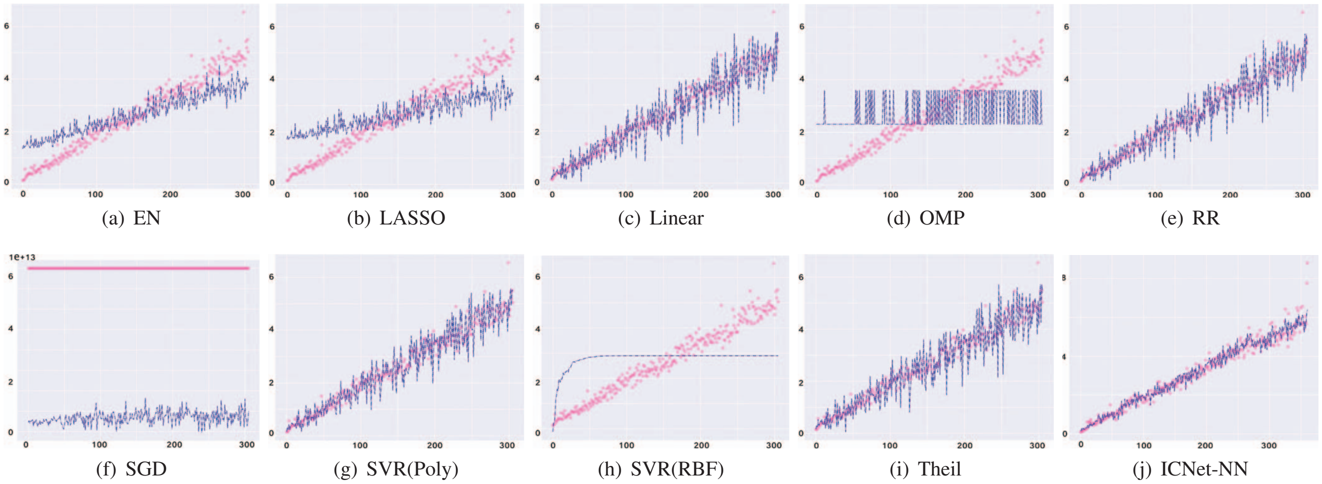
Next, Fig. 3 illustrates several predicted value along with real value to analyze the prediction characterization. Since there is little difference in dataset 2, we choose several competitive baselines in dataset 1 experiments under all feature setting. Several baselines performed very badly such as OMP and SGD which only output values around a constant level. SVR (RBF) is also bad and yield constant value when the real runtime is larger than a threshold. The results of EN and LASSO is positively related to the real values, but the correlation parameters are significantly different from the truth. Linear, RR, SVR (POLY) and Theil predicted the values that are relatively closer than that of the other baselines, but with high variance.

D. Case Study: Attentions on Attributes

The subsection studies the attention mechanism quantitatively. Several circuits are evaluated, as shown in Table III. Gate number consistently attracted greater attention than the gate type by 9.64% on average. This motivates us to study the correlation between actual runtime and gate number. The Pearson (P) and Spearman (S) correlation are 0.8238 and 0.9722 on average in Table III. Take circuit c7553 as an example, the runtime is 2.37% of gate number. Different circuits show different linear parameters, which gives us a convenient message that can accurately predict the deobfuscation time and could serve circuit obfuscation task.

Table III: Case study: attributes and extracted rules.

circuit	gate #	gate type	corr(P/S)	linear param
c7553	56.40%	43.59%	0.8754 / 0.9345	0.0237
c499	54.39%	47.05%	0.8149 / 0.9965	0.1300
c2670	52.94%	47.05%	0.7769 / 0.9753	0.0559
c1335	56.27%	43.72%	0.8282 / 0.9846	0.0599



(a) EN (b) LASSO (c) Linear (d) OMP (e) RR

(f) SGD (g) SVR(Poly) (h) SVR(RBF) (i) Theil (j) ICNet-NN

Figure 3: Comparison between predictions and real values: Pink dot are real values, blue lines are the predictions. x-axis is data index in testing data while y-axis is runtime value in log scale. Note that only SGD has different y-axis scale, i.e., 1e+13.

V. Conclusion

In this work, we have introduced a neural network model for recovering SAT runtime on ICs, which expedites the evaluation on the hardness of obfuscated instances and therefore boosts the efficiency of developing obfuscation policy. To properly fuse graph structure and gate features, an enhanced graph convolutional operator is introduced. The proposed ICNet can avoid attribute propagation which is in the original GCN but not suitable for ICs. ICNet automatically extracts determinant features and aggregates gate representation regarding the runtime. Experiments on real-world datasets suggest that the proposed model is capable of modelling the runtime regarding the circuit graph accurately and stably, improving the baselines by a significant margin.

References

[1] C. M. Bishop and T. M. Mitchell. Pattern recognition and machine learning. 2014.

[2] F. R. Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997.

[3] X. Dang, H. Peng, X. Wang, and H. Zhang. Theil-sen estimators in a multiple linear regression model. *Olemiss. edu*, 2008.

[4] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, pages 3844–3852, 2016.

[5] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

[6] D. K. Hammond, P. Vandergheynst, and R. Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[7] S. Khaleghi and W. Rao. Hardware obfuscation using strong pufs. In *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 321–326. IEEE, 2018.

[8] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[9] G. Kolhe and et.al. On custom lut-based obfuscation. In *Great Lakes Symposium on VLSI*, 2019.

[10] G. Kolhe and et.al. Security and complexity analysis of lut-based obfuscation: From blueprint to reality. In *Int. Conference On Computer Aided Design*, 2019.

[11] Q. V. Le, J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, and A. Y. Ng. On optimization methods for deep learning. In *ICML*, pages 265–272. Omnipress, 2011.

[12] M. Lechner and et al. ResCoNN: Resource-efficient FPGA-accelerated CNN for traffic sign classification. In *IEEE Int. Green and Sustainable Computing Conference (IGSC)*, 2019.

[13] Q. Li, Z. Han, and X.-M. Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[14] D. Liu, C. Yu, X. Zhang, and D. Holcomb. Oracle-guided incremental sat solving to reverse engineer camouflaged logic circuits. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016*, pages 433–438. IEEE, 2016.

[15] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[16] I. T. P. R. T. . most counterfeited parts represent a $169 billion potential challenge for global semiconductor industry. https://technology.ihs.com/405654/top5-most-counterfeited-parts-represent-a-169-billion-potentialchallenge-for-global-semiconductor-market,2.

[17] A. Y. Ng. Feature selection, l 1 vs. l 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

[18] S. Roshanisefat, H. Mardani Kamali, and A. Sasan. Srclock: Sat-resistant cyclic logic locking for protecting the hardware. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, GLSVLSI '18, 2018.

[19] D. Selsam, M. Lamm, B. Bünz, P. Liang, L. de Moura, and D. L. Dill. Learning a SAT solver from single-bit supervision. *ArXiv*, abs/1802.03685, 2018.

[20] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[21] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[22] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004.

[23] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, 2015.

[24] P. Subramanyan, S. Ray, and S. Malik. Evaluating the security of logic encryption algorithms. In *IEEE Int. Symp. on Hardware Oriented Security and Trust (HOST)*, 2015.

[25] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *ICLR*, 2018.

[27] S. Wang, G. Ananthanarayanan, Y. Zeng, N. Goel, A. Pathania, and T. Mitra. High-throughput CNN inference on embedded ARM big.little multi-core processors. *CoRR*, abs/1903.05898, 2019.

[28] Y. Xie and A. Srivastava. Delay locking: Security enhancement of logic locking against ic counterfeiting and overproduction. In *ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2017.

[29] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri. On improving the security of logic locking. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 35(9):1411–1424, Sept 2016.

[30] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu. Provably-secure logic locking: From theory to practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1601–1618. ACM, 2017.

[31] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan. SMT-attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks. In *Transaction of Cryptography Hardware and Embedded Systems*, 2019.

[32] H. Zhou, R. Jiang, and S. Kong. Cycsat: Sat-based attack on cyclic logic encryptions. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 49–56, Nov 2017.

[33] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.