IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

A Deep Learning Approach for Active S-box Prediction of Lightweight Generalized Feistel Block Ciphers

MOHAMED FADL IDRIS[1], JE SEN TEH[1], JASY LIEW SUET YAN[1], AND WEI-ZHU YEOH[2].
[2]CISPA Helmholtz Center for Information Security, Saarland University
[1]School of Computer Sciences, Universiti Sains Malaysia

Corresponding author: Je Sen Teh (e-mail: jesen_teh@usm.my)

This work is supported in part by the Ministry of Education Malaysia under the Fundamental Research Grant Scheme (FRGS), project number FRGS/1/2019/ICT05/USM/02/1.

ABSTRACT One of the main security requirements for symmetric-key block ciphers is resistance against differential cryptanalysis. This is commonly assessed by counting the number of active substitution boxes (S-boxes) using search algorithms or mathematical solvers that incur high computational costs. These costs increase exponentially with respect to block cipher size and rounds, quickly becoming inhibitive. Conventional S-box enumeration methods also require niche cryptographic knowledge to perform. In this paper, we overcome these problems by proposing a data-driven approach using deep neural networks to predict the number of active S-boxes. Our approach trades off exactness for real-time efficiency as the bulk of computational work is brought over to pre-processing (training). Active S-box prediction is framed as a regression task whereby neural networks are trained using features such as input and output differences, number of rounds, and permutation pattern. We first investigate the feasibility of the proposed approach by applying it on a reduced (4-branch) generalized Feistel structure (GFS) cipher. Apart from optimizing a neural network architecture for the task, we also explore the impact of each feature and its representation on prediction error. We then extend the idea to 64-bit GFS ciphers by first training neural networks using data from five different ciphers before using them to predict the number of active S-boxes for TWINE, a lightweight block cipher. The best performing model achieved the lowest root mean square error of 1.62 and R^2 of 0.87, depicting the feasibility of the proposed approach.

INDEX TERMS Active S-boxes, block cipher, cryptanalysis, deep learning, differential cryptanalysis, lightweight cryptography, neural networks, TWINE

I. INTRODUCTION

BLOCK ciphers are ubiquitous cryptographic primitives that not only provide data confidentiality but are used as building blocks for myriad other cryptographic algorithms and protocols. More recently, lightweight block ciphers have been gaining popularity as they provide data security to resource-constrained devices such as RFID tags or smart Internet-of-Things (IoT) devices [1]. A block cipher processes a fixed-length (block) of data using a key-dependent transformation that usually consists of generic operations such as substitution and permutation. This key-dependent transformation will be applied multiple times (multiple rounds) before the final ciphertext is produced. Keys for each encryption round, known as round keys, are derived from a master key using a key scheduling algorithm. Depending on their underlying structure, modern block ciphers can be classified into different categories such as the generalized Feistel structure (GFS), addition-XOR-rotate (ARX), and substitution-permutation network (SPN). A block cipher is considered to be secure enough for practical applications if it has shown to be resistant against various state-of-the-art cryptanalysis techniques over a certain period of time.

In 1990, Biham and Shamir introduced differential cryptanalysis, a statistical tool that observes the propagation of plaintext differences (input differences) through a block cipher to produce ciphertext differences (output differences) [2]. Even today, resistance against differential cryptanalysis is considered one of the de facto requirements for block ciphers, and more recently, authenticated ciphers. It has even been used to analyse the security of image encryption algo-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

rithms [3]. An input difference is defined as

$$\Delta X = X' \oplus X'', \tag{1}$$

where X' and X'' are two distinct plaintexts and $\oplus$ refers to the exclusive-OR (XOR) operation. An output difference, ΔY can be similarly defined using a pair of corresponding ciphertexts, Y' and Y''. We denote an r-round differential path or differential trail as the propagation of ΔX to ΔY after r rounds of encryption, $\Delta X \xrightarrow{r} \Delta Y$. Each differential path holds with a certain probability, $Pr(\Delta X \xrightarrow{r} \Delta Y) = 2^{-p}$ which an adversary wishes to maximize. Generally, a b-bit block cipher is considered insecure if $p < b$ because the differential trail can be used as a statistical distinguisher for key recovery attacks.

Many block cipher designs that are based on substitution boxes (S-boxes) estimate their security margins against differential cryptanalysis by counting the number of active S-boxes [4]–[7]. An S-box is considered differentially active when it receives a nonzero difference as an input. For each nonzero input, there are several possible output differences with varying chances of occurrence. Thus, the difference propagation through each active S-box incurs a statistical cost, lowering the overall differential probability by a factor of p_{sbox} which is the probability that a particular difference propagation can occur. p_{sbox} is obtained by deriving a differential distribution table for a particular S-box. Resistance is estimated using the worst-case probability (from the perspective of the block cipher designer), which for optimal 4 and 8-bit S-boxes are usually $p_{sbox} = 2^{-2}$ and $p_{sbox} = 2^{-6}$ respectively. The overall differential probability for a differential trail is then calculated as $Pr(\Delta X \xrightarrow{r} \Delta Y) = (p_{sbox})^{AS}$, where AS is the total number of active S-boxes after r rounds of the cipher. In other words, each active S-box incurs a penalty, lowering the overall differential probability. Thus, the goal of a block cipher designer is to maximize the number of active S-boxes in their block cipher designs whereas a cryptanalyst aims to find a differential trail that minimizes this number. Identification of the best differential trail or minimum of number of active S-boxes can be performed using search algorithms that are computationally intensive [8], [9] or by representing it as Boolean satisfiability (SAT) or mixed-integer linear programming (MILP) problems that are then solved computationally [10], [11]. These approaches usually require considerable cryptographic knowledge to execute efficiently.

Recently, concepts from differential cryptanalysis have been used for an entirely different purpose - to train machine learning algorithms for cryptanalysis applications. The relationship between machine learning and cryptanalysis was examined nearly two decades ago by Rivest [12] but recently began to gain traction thanks to Gohr who introduced the first successful application of machine learning (or more specifically, deep learning) in the context of classical cryptanalysis [13]. A machine learning-based differential distinguisher trained using differential data was used to achieve the best cryptanalytic attack to date on a round-reduced Speck32/64. His findings showed that machine learning distinguishers were superior to conventional differential distinguishers, paving the way for future research in the area.

A. CONTRIBUTION

In this paper, we train deep neural network models to predict the number of active S-boxes for lightweight block ciphers based on general block cipher features and differential data. More specifically, the training samples consist of features such as truncated input and output differences, the number of rounds, and permutation pattern whereas the number of active S-boxes is used as labels. We modified the branch-and-bound search from [8] to generate randomly sampled data for training and testing[1]. Unlike most of the existing machine learning-based cryptanalysis methods, ours is generalizable to more than just one block cipher; the resulting neural network can predict the number of active S-boxes for block ciphers other than the ones it has been trained for.

We first investigate the feasibility of the approach on smaller-scale (4-branch) GFS ciphers. This allows us to identify optimal neural network models for the prediction task within a practical amount of time. In addition, we explore the impact of each block cipher feature and its representation on prediction error. Compared to other neural networks such as long short-term memory (LSTM) and convolutional neural networks (CNN), we found that fully connected neural networks were best suited for the prediction task due to the innate complexity of block ciphers. The best performing models achieved low root mean square error (RMSE) values ranging between 1.67 to 1.96 and R^2 scores of approximately 0.88. Our results show that the permutation pattern has the biggest impact on prediction performance as its removal leads to the largest increase in prediction errors (88.8%), followed by input differences (23%) and finally output differences (15.8%). We also found that splitting these key features into multiple features (e.g. representing each bit in a truncated difference as separate features) leads to improved prediction performance.

Next, we extend the idea towards full-scale (16-branch or 64-bit) lightweight block ciphers. Here, we investigate the capability of trained neural network models to generalize to a block cipher that they have not seen during training. The lightweight GFS block cipher, TWINE deveoped by NEC Corp. [6] was used as the target cipher. The neural network model was trained using data generated from five GFS ciphers, each with different permutation patterns and limited to eight rounds. In the baseline experiments, this model was trained and tested using data taken from these GFS ciphers. Then, we use this data to train deep learning models to label data samples taken from a round-reduced TWINE. The best performing model achieved the lowest RMSE of 1.62 (as compared to the baseline result of 0.71) and R^2 of 0.87 (a 12.6% decrease from the baseline result). From the

[1]Supplementary code for this paper, including the search algorithm, is available at https://github.com/jesenteh/deeplearning-gfs.

perspective of a cryptanalyst, the trained deep learning model can be expected to correctly predict the number of active S-boxes that deviates, on average, by ± 1.62 from the actual result. These results showcase the feasibility of the proposed approach that has a myriad of practical applications, ranging from a rapid assessment of block cipher security to filtering differential pairs for cryptanalytic attacks. It is also a data-driven approach that can be easily adopted without requiring niche cryptographic expertise.

B. OUTLINE

The rest of this paper is structured as follows. Related work, background information on neural networks, truncated differentials, and GFS are provided in Section II. The experiments on both the 4-branch and 16-branch ciphers are described in Section III followed by experimental results in Section IV. Section V discusses the important findings and potential applications of the proposed work before the paper is concluded in Section VI.

II. BACKGROUND

A. RELATED WORK

The use of machine learning in cryptography may not be new but successful applications in cryptanalysis have only recently begun to emerge. One popular use case of machine learning is to distinguish or identify encryption algorithms based on ciphertext data. These approaches have been used to analyze data encrypted by popular block ciphers such as AES, 3DES, Blowfish and Camellia [14]–[17]. Cryptanalysts have also attempted to use machine learning in a straightforward manner - performing decryption of ciphertexts without knowledge of the secret key. This is equivalent to training a machine learning model to emulate or mimic an encryption algorithm for a fixed secret key. For this purpose, [18] utilized unsupervised learning with neural networks to attack classical ciphers such as the Vigenere and Shift ciphers.

Mishra et al. investigated whether neural networks can be used to predict plaintext bits of the block cipher PRESENT based on data obtained from any particular round [19]. They used the same approach in [20] against the FeW cipher, both with limited success. Xiao et al. described a black-box security evaluation approach to measure the strength of proprietary ciphers without knowledge of the encryption algorithms themselves [21]. They quantified the strength of a cipher by measuring how difficult it was for a neural network to mimic the cipher algorithm. Their results showed that the security of Hitag2 (a cipher used for keyless entries in modern cars) was weaker than 3 rounds of the Data Encryption Standard (DES). Cipher mimicking has also been explored in [22]. The researchers optimized a deep neural network to decrypt 64-bit DES ciphertexts without knowledge of the secret key, achieving an accuracy of up to 90%.

Perov demonstrated that using machine-learning techniques, it was possible to distinguish the ciphertexts of round-reduced ciphers from random sequences [23]. This was achieved with a smaller sample size (2^{12} bits) as compared to conventional statistical methods. A machine learning model was used in [24] to predict the outputs of a quantum random number generator. Their model could perform predictions better than random guessing even when a given random sequence passed the NIST statistical test. Machine learning was also used to identify locations that are sensitive to fault attacks in block and stream ciphers [25].

Although there have been attempts in the past to identify differential characteristics using neural networks [26], Gohr introduced what was considered the first successful application of machine learning from the perspective of conventional cryptanalysis back in 2019 [13]. A machine learning-based differential distinguisher was developed and used to attack Speck32/64 reduced to 11 rounds. The machine learning distinguishers were found to be superior to conventional differential distinguishers. [27] expanded upon this concept by developing deep learning distinguishers based on all-in-one differentials for non-Markov ciphers. As proofs-of-work, they performed distinguishing attacks using 8-round distinguishers on Gimli-Hash and Gimli-Cipher, each with trivial complexity. Machine learning has also been applied to perform linear cryptanalysis on DES based on plaintext-ciphertext pairs and linear expressions [28]. [29] recently proposed machine learning classifiers that can classify differential trails as secure or insecure based on differential data. Proof-of-concept experiments were performed on small-scale GFS ciphers. Their findings showed that the trained models were able to generalize to ciphers that the models have not seen before. Last but not least, [30] attempted key recovery attacks on block ciphers, Simon and Speck, using deep learning. They were successful in recovering encryption keys for full-round Simon32/64 and Speck32/64 only when the keyspace was restricted to text-based keys.

B. NEURAL NETWORKS

A neural network is a machine learning algorithm that can train a computer to perform a task from training examples. Modeled loosely on how the human brain works, a neural network composes of connected processing units called neurons. The neurons are organized into multiple layers of interconnected nodes, including an input layer, output layer, and one or more hidden layers. The more hidden layers they are, the “deeper" the neural network. Each neuron is assigned a weight, which is multiplied by the data from every incoming connection. The resulting products for a neuron are then added together. If the sum of products reaches above a threshold, the neuron “fires" and sends along the number to all its outgoing connections [31].

Training a neural network involves optimizing these weights to minimize the difference between the actual (expected) value and a predicted value, otherwise known as the loss function. Given a set of initial weights, training data enters the input layer and is transformed as it passes successively through each layer of the neural network until it reaches the output layer that determines the result of the prediction. The weights are continuously adjusted during

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

training until the loss function reaches near-zero error.

Loss functions that were used in the proposed work include:

- **Root mean square error (RMSE)** is the square root of the average of squared differences between an actual value and its predicted value (prediction error) for all instances in a dataset.
- **Mean absolute error (MAE)** calculates the average of the absolute values of prediction errors for all instances in a dataset.
- The **quantile loss function** is applied to predict intervals or range of predictions rather than just single point predictions. The value below which a fraction of predictions in a group falls is known as a quantile.

We have selected to experiment with the fully connected deep neural network architecture in our regression task. After experimenting with various neural network architectures such as LSTM and CNN, we found that a fully connected neural network performed consistently for the active S-box prediction problem. In addition to shorter training times (under an hour for the fully connected neural network as compared to several hours or more for CNN and LSTM when training with 100 epochs for 4-branch GFS), fully connected neural networks were also found to be more accurate for the given problem. Preliminary testing showed that the fully connected neural networks could achieve RMSE values of under 2.0, while CNN and LSTM both achieved RMSE values upward of 2.0. In addition, a fully connected neural network does not make any assumption to the input, thus making it flexible to be applied in our problem [32]. A fully connected neural network consists of a series of fully connected layers, where each neuron is connected to all neurons in the following layer. Figure 1 illustrates the fully connected neural network that was used in our experiments on the 4-branch GFS ciphers, where the edges represent the weights of each input. A fully connected neural network with many layers is also referred to as a deep network.

C. TRUNCATED DIFFERENTIAL CRYPTANALYSIS

Differential cryptanalysis relies on identifying an r-round differential trail, $\Delta X \xrightarrow{r} \Delta Y$ with a high probability of occurring. An S-box is considered active when it receives a non-zero input difference, which will be mapped to several possible non-zero differences depending on the actual value of the round key, rk_r. An example of an active S-box is shown in Figure 2, whereby the left S-box is active while the right S-box is inactive. Truncated differential cryptanalysis is a generalization of differential cryptanalysis whereby differences that are only partially determined can be used in attacks. In our paper, we adopt the same definition as [33], whereby a full input difference or concrete differential state, ΔX can be mapped to a truncated differential state $\Delta \hat{X}$ by dividing the input difference into t-bit blocks that have the same size as the S-box. For example, a 64-bit input difference can be truncated to a 16-bit truncated difference if

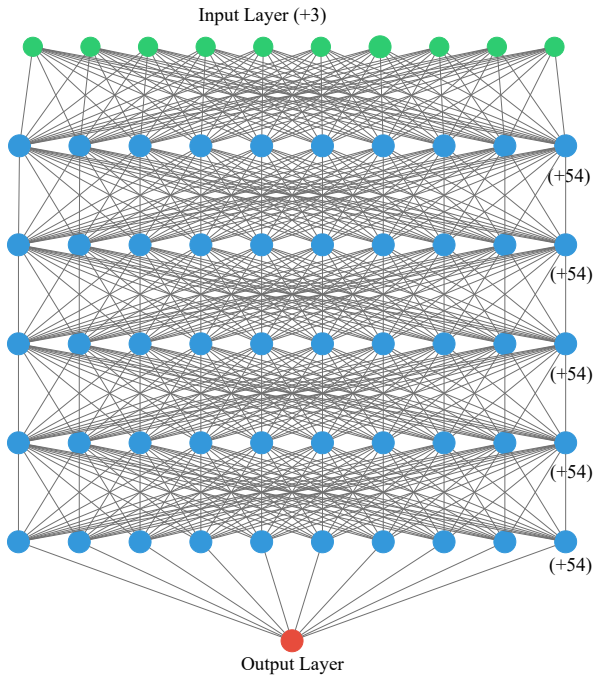


FIGURE 1: Fully Connected Neural Network

4-bit ($t = 4$) S-boxes are used. As long as any of the t bits are nonzero in the concrete differential state, its corresponding truncated bit is set to 1. Otherwise, if a t-bit block in the concrete differential state is zero, its corresponding truncated bit is set to 0. An example of the truncation process is depicted in Figure 2.

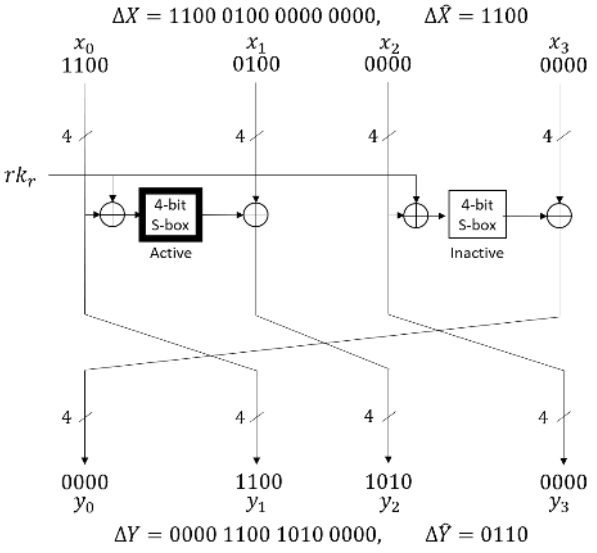


FIGURE 2: 4-branch GFS (1 round)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

D. GENERALIZED FEISTEL STRUCTURE AND TWINE

One of the common structures for a block cipher is GFS, a generalization of the Feistel structure made popular by DES. A GFS divides an input into b sub-blocks (or words), where $b \geq 2$. The size of these sub-blocks is usually dependent on the S-box size. The Type-II GFS [34], [35] is one of the more popular GFS variants used in many block cipher designs where one sub-block from each pair of sub-blocks will undergo a key-dependent transformation (referred to as a round function) before being XOR-ed with the other half. This is followed by a permutation layer. Figure 2 illustrates one round of a 4-branch Type-II GFS cipher that will be used in the proof-of-concept experiments in our proposed work. In the diagram, an example of an input to output difference propagation is provided, along with their truncated representation. The left S-box is considered *active* because it receives a nonzero difference as an input. The round function, F is defined as

$$F(x_i) = s(x_i \oplus rk_i), \tag{2}$$

where s is the S-box function and rk_i is the round key. In essence, this 4-branch structure can represent either a 16 or 32-bit block cipher depending on the S-box size. There are a total of 4! = 24 possible permutation patterns for a 4-branch permutation. The permutation pattern depicted in Figure 2 is $P = \{1, 2, 3, 0\}$. The structure of the block cipher TWINE [6] is basically an extension of this 4-branch GFS to 16 branches, with a permutation pattern of $P = \{5, 0, 1, 4, 7, 12, 3, 8, 13, 6, 9, 2, 15, 10, 11, 14\}$. TWINE is a lightweight 64-bit block cipher that supports 80 and 128-bit keys designed for hardware efficiency. It processes the 64-bit plaintext as 16 4-bit sub-blocks. The size of the sub-blocks corresponds to the cipher's S-box size. TWINE will be used as the target cipher for our generalization experiments.

III. METHODOLOGY

A. PROBLEM FRAMING

The overall goal of the proposed work is to train neural network models to predict the number of active S-boxes of a GFS block cipher based on features such as truncated differences, the number of rounds, and permutation patterns. Using supervised learning, we framed the problem as a regression task because the goal is to predict the number of active S-boxes consisting of non-negative integers. By training the models with sufficient data samples, they will eventually learn the relationship between the cipher features and the number of active S-boxes, and be able to predict them with reasonable accuracy. Experiments were performed for both the 4-branch and 16-branch GFS ciphers.

The minimum number of active S-boxes is 0 when the plaintext consists entirely of zeroes whereas the maximum number of active S-boxes, AS_{max} will vary depending on the number of rounds, r, block size, b, S-box size, t, and underlying structure of a block cipher. A generic GFS block cipher such as TWINE would have a theoretical maximum of $AS_{max} = \frac{b \times r}{2t}$ active S-boxes whereas a classical SPN block cipher such as PRESENT [4] would have $AS_{max} = \frac{b \times r}{t}$. The goal of a cryptanalyst is to identify a differential trail that minimizes the number of active S-boxes so it can be used in attacks against the cipher.

RMSE and R-squared (R^2) are two performance metrics selected to evaluate the performance of the neural networks on the test set. RMSE measures the magnitude of prediction errors made by a machine learning model. Ideally, a trained model should have an RMSE value of as close to zero as possible. RMSE indicates the average deviation of the predicted number of active S-boxes from the actual one. R^2 is the quantitative measure of the goodness of fit of a regression model. In other words, it represents how closely prediction values conform to the best-fitted regression line. The R^2 metric ranges between 0 and 1, where 1 is the ideal value. Both metrics will be used to determine how accurate the trained models are in predicting the number of active S-boxes.

B. NEURAL NETWORK ARCHITECTURE

In all of the following experiments, we utilized fully connected neural networks. We first performed hyperparameter tuning[2] to identify the optimal number of layers (2-16), number of neurons per layer (16-1024), loss function, optimizer, number of epochs (100-1000), and batch size (8-1024) for the regression task. Based on our experiments, we selected a neural network with four hidden layers. The number of neurons per layer differs depending on the target block cipher. For 4-branch GFS ciphers, there are 13 neurons in the input layer (equivalent to the number of input features), four hidden layers with 64 neurons each, and an output layer with 1 neuron to represent the predicted number of active S-boxes as depicted in Figure 1. As for 16-branch GFS ciphers, there are 49 neurons for the input layer, 512 neurons for the hidden layers, and 1 neuron for the output layer. The remaining hyperparameters used in our experiments are summarized below:

- 4-branch GFS Experiments
  - -- Optimizer: Adam with 0.001 learning rate
  - -- Activation Function for Hidden Layers: Exponential linear unit (ELU)
  - -- Activation Function for Output Layer: Linear Activation Function
  - -- Epochs: 100 epochs
  - -- Batch Size: 32
- 16-branch GFS Experiments
  - -- Optimizer: Adam with 0.001 learning rate
  - -- Activation Function for Hidden Layers: ELU
  - -- Activation Function for Output Layer: Linear Activation Function
  - -- Epochs : 200 epochs
  - -- Batch Size: 32

[2]Values in brackets indicate the range being tested during hyperparameter tuning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

In the experiments on full-scale GFS ciphers, we have selected regularization as a means to avoid overfitting. Regularization involves adding an additional penalty term to the loss function. This additional term controls or regulates the function by placing constraints on the amount and type of information being stored by the model, forcing it to focus more on prominent patterns. This leads to higher chances of the model generalizing well to unseen data.

C. DATA PREPARATION

By using the smaller scale 4-branch GFS ciphers for proof-of-concept experiments, we could generate a large dataset within a practical amount of time. In addition, data for all 24 possible permutation patterns (each of which can be viewed as an individual 4-branch GFS cipher) could be generated. We generated a dataset of 500000 samples (denoted as $DATA_{4b}$) using a modified branch-and-bound differential search algorithm [8]. The original differential search algorithm was a depth-first search algorithm that explores all possible differential trails going through a cipher, bounding those that will not lead to improved results in terms of differential probability. Modification of the original search algorithm was required because our goal was not to identify a differential trail with optimal probability nor the minimum number of active S-boxes but rather to enumerate every possible branch to ensure that the dataset consisted of randomly sampled differential data from round 1 to 12 of the 24 GFS ciphers. This was performed by removing all bounding criteria and differential probability calculations, focusing only on counting the number of active S-boxes for each differential trail and all its possible branches. The format of the data samples is depicted in Table 1.

$\Delta\hat{X}$	$\Delta\hat{Y}$	Active S-boxes	r	P
1101	1101	20	12	2301
1011	0011	9	10	0213
1101	0101	11	9	1203
0101	1101	17	11	1203
0111	0101	6	7	0213

TABLE 1: Examples of data samples for the 4-branch GFS

As it was not practical to enumerate every possible permutation pattern or block cipher variant for a 16-branch GFS, we selected only six for our experiments. The first was TWINE itself, our target cipher, whereas the other five were based on permutation patterns with the best cryptographic properties based on findings in [35]. Notably, they all achieve full diffusion in eight rounds and have a minimum number of active S-boxes of at least 40 after 20 rounds. All six permutation patterns are listed in Table 2.

The dataset was generated using the same modified branch-and-bound search. 100000 data samples were generated for each cipher, consisting of 12500 randomly selected samples from round 1 to 8. Apart from being able to generate data samples within a practical amount of time, the 8-round limit was selected because it is the minimum number of rounds required to achieve full diffusion of plaintext bits

Name	Permutation Pattern, P
No. 5	5,2,9,4,11,6,15,8,3,12,1,10,7,0,13,14
No. 7	1,2,11,4,3,6,7,8,15,12,5,14,9,0,13,10
No. 9	1,2,11,4,9,6,15,8,5,12,7,14,3,0,13,10
No. 10	7,2,13,4,11,8,3,6,15,0,9,10,1,14,5,12
No. 12	1,2,11,4,15,8,3,6,7,0,9,12,5,14,13,10
TWINE	5,0,1,4,7,12,3,8,13,6,9,2,15,10,11,14

TABLE 2: 16-branch permutation patterns

[35]. We limited the number of output differences per input difference to four, to ensure that there was a larger variety of input differences in the dataset. The format of the data samples is similar to the one depicted in Table 1. The 500000-sample dataset that consists of the five GFS ciphers (excluding TWINE) is denoted as $DATA_{16b}$ whereas the 100000-sample dataset generated from only TWINE is denoted as $DATA_{TW}$. Our overall methodology shown in Figure 3 is further detailed in the upcoming subsections.

D. ACTIVE S-BOX PREDICTION FOR 4-BRANCH GFS

Our experiments on the 4-branch GFS were divided into two main phases: In Phase 1-1, we identified the best performing neural network architecture to become our baseline model whereas in Phase 1-2, we examined the effect of excluding certain features on prediction error. Prediction error was measured using RMSE and R^2. The $DATA_{4b}$ dataset was used in both phases. All experiments were implemented using Tensorflow 2.3.1 on an Intel Core i5 Processor, 8GB RAM using Python and Jupyter Notebook.

1) Phase 1-1 - Baseline Experiment

The first experiment involved training a baseline model using all the available block cipher features, where a train-test split with a ratio of 80:20 was applied to the $DATA_{4b}$ dataset (400000 training samples, 100000 test samples). In terms of feature representation, we first performed our experiments with each feature being represented as individual variables as illustrated in Table 1. However, we later found that the trained models performed better when the truncated difference and permutation features were split into four separate variables each (four features) as shown in Table 3. In this paper, we only report results from experiments that used this alternative representation. We trained three separate deep learning models, each using a different loss function (MAE, RMSE, and quantile), to determine which one led to more accurate predictions.

$\Delta\hat{X}$	$\Delta\hat{Y}$	Active S-boxes	r	P
1, 1, 0, 1	1, 1, 0, 1	20	12	2, 3, 0, 1
1, 0, 1, 1	0, 0, 1, 1	9	10	0, 2, 1, 3
1, 1, 0, 1	0, 1, 0, 1	11	9	1, 2, 0, 3
0, 1, 0, 1	1, 1, 0, 1	17	11	1, 2, 0, 3
0, 1, 1, 1	0, 1, 0, 1	6	7	0, 2, 1, 3

TABLE 3: Examples of data samples for the 4-branch GFS with alternate representation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

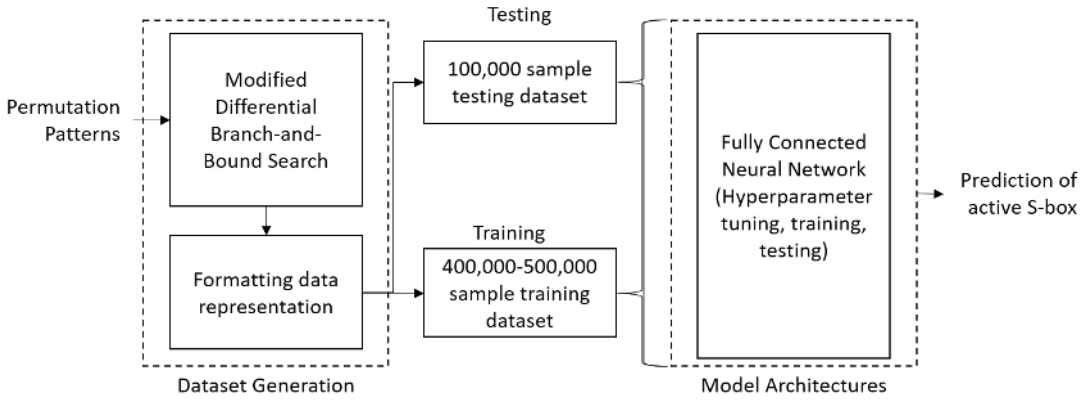


FIGURE 3: Methodology for GFS Experiments

2) Phase 1-2 - Feature Refinement

In this phase, we explored the effect of specific block cipher features on prediction error. This was performed by excluding only one of the features (truncated output difference, truncated input difference, or permutation) then using the remaining ones for training. We note that r was not removed a feature because the number of active S-boxes in a block cipher is directly dependent on it. The experiment was performed three times (once for each feature). The magnitude of performance degradation would correspond to the impact that a particular block cipher feature has on the prediction task. The best performing model from Phase 1-1 was used for training and comparison purposes.

E. ACTIVE S-BOX PREDICTION FOR 16-BRANCH GFS

Our experiments on 16-branch GFS ciphers were also divided into two main phases: In Phase 2-1, we identified the best performing neural network architecture to become our baseline model whereas in Phase 2-2, we determined if deep learning models trained using data from five GFS ciphers can generalize to data from an unseen GFS cipher, TWINE. This showcases the improved flexibility of the proposed method as compared to existing machine learning-based cryptanalysis methods that are cipher-specific [13], [27]. Prediction error was again measured using RMSE and R^2. The same computing setup as the 4-branch GFS experiments (Tensorflow 2.3.1, Intel Core i5 Processor, 8GB RAM, Python, Jupyter Notebook) was used in Phase 2 as well.

1) Phase 2-1 - Baseline Experiment

In this phase, the models were trained and tested using the $DATA_{16b}$ dataset. A train-test split with a ratio of 80:20 was used (400000 training samples, 100000 test samples). In terms of feature representation, we split the truncated input and output differences, and permutation pattern into 16 separate features. In total, 49 features were used for training (corresponding to the 49 neurons in the input layer of the deep neural network). We also investigated the use of three different loss functions (MAE, RMSE, quantile) to determine their impact on prediction error.

2) Phase 2-2 - Generalization Experiment

Next, we determined if the proposed deep learning models were capable of labeling unseen data or in other words, generalize to a block cipher they have never encountered before. The models were trained using 500000 data samples from $DATA_{16b}$ and tested using the 100000 samples from $DATA_{TW}$. By using separate datasets for training and testing, the deep learning models would have never encountered data samples from TWINE during the training stage. This experiment represents one of the practical use cases of the proposed approach, whereby a deep learning model can be pre-trained using available data, and then used to perform active S-box predictions for other block ciphers in the future. The same fully connected neural network architecture and hyperparameters as the baseline model was used for these experiments. However, rather than just one loss function, we still examined the use of all three (MAE, RMSE, and quantile). In addition, we used the L2 regularizer (regularizers.l2(0.001) in TensorFlow) to overcome overfitting problems for better prediction performance.

IV. EXPERIMENTAL RESULTS

A. 4-BRANCH GFS RESULTS

The experimental results for Phase 1-1 are shown in Table 4 which compares the baseline performance of the deep learning models based on three loss functions, MAE, RMSE, and quantile. Overall, all models could predict the number of active S-boxes with an RMSE of 1.96 and under. This implies that on average, predictions vary from the real value by approximately two active S-boxes. The R^2 values of 0.88 to 0.89 imply that there is a strong relationship between the actual and predicted values for all three cases. The quantitative metrics suggest that the use of any of the three loss functions will lead to similar performance.

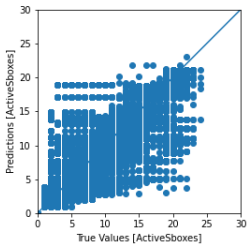
From the perspective of differential cryptanalysis, an average error of two 4-bit active S-boxes translates to a probability penalty of $2^{-2\times AS} = 2^{-4}$. This penalty is more sig-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

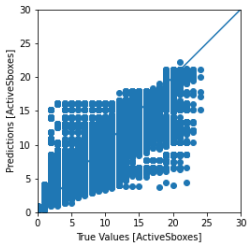
nificant for a smaller number of rounds because the number of active S-boxes is generally lower. Thus, the deep learning models can also be compared in terms of how well they perform for a smaller number of block cipher rounds which generally have fewer active S-boxes. Figure 4 illustrates that the quantile loss function has slightly better curve fitting as compared to the other loss functions, especially when the number of active S-boxes is smaller. Therefore, the neural network model with the quantile loss function was chosen as the baseline model for Phase 1-2.

Loss Function	RMSE	R^2
MAE	1.67	0.88
RMSE	1.72	0.89
Quantile(90%)	1.96	0.88

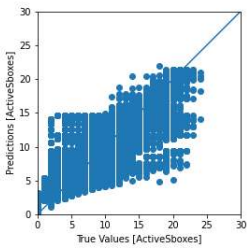
TABLE 4: Predicted number of active S-boxes based on loss functions (4-branch GFS)



(a) Prediction error for MAE loss function



(b) Prediction error for RMSE loss function



(c) Prediction error for quantile loss function

FIGURE 4: Comparison of loss functions for baseline experiments (4-branch GFS)

The experimental results for Phase 1-2 are shown in Table 5. Generally, exclusion of either the truncated input or output difference had a minor negative effect on prediction error. Experimental evidence from Table 5 suggests that the input difference, $\Delta\hat{X}$ plays a slightly bigger role in prediction performance than the output difference, $\Delta\hat{Y}$. Removal of $\Delta\hat{X}$ led to a 23% increase in prediction errors (as compared to the baseline results) whereas removal of $\Delta\hat{Y}$ led to a 15.8% increase. In both cases, R^2 values only reduced by an average of 6.3% as compared to the baseline, implying that there was still a strong relationship between actual and predicted values. In contrast, excluding the permutation pattern had a detrimental impact on the model's performance as it led to a significant increase of 88.8% in prediction errors (RMSE), with the R^2 score dropping by 31.8%. By excluding the permutation pattern from the training process, a neural network must learn how to differentiate between 24 different block ciphers based entirely on the input and output truncated differences. This is a complex task due to the strong diffusive property of block ciphers. Our experimental results confirmed this notion by showing that the deep learning model relied heavily on permutation patterns to make accurate predictions. Based on our findings, all three features were included in the 16-branch GFS experiments to ensure optimal prediction performance.

Model	RMSE	R^2
All features (baseline)	**1.96**	**0.88**
Only $\Delta\hat{Y}$ removed	2.27	0.83
Only $\Delta\hat{X}$ removed	2.41	0.82
Only P removed	3.7	0.60

TABLE 5: Predicted number of active S-boxes when removing specific features

B. 16-BRANCH GFS RESULTS

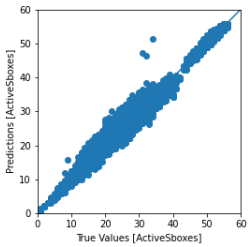
The baseline performance of the deep learning models when performing predictions for 16-branch GFS ciphers are summarized in Table 6. Generally, the models had low prediction errors for all three loss functions, achieving RMSE values of 0.96 and under. From a cryptanalyst's perspective, the results imply that the proposed models can perform active S-box prediction with low error (± 1 active S-box on average) for block ciphers that the model has seen. The R^2 values are also near-ideal (ranging between 0.992 to 0.994), depicting a strong relationship between the predicted and actual values. We note that there is a significant improvement in prediction performance for the 16-branch GFS ciphers as compared to 4-branch ciphers, which could be attributed to the larger number of training features (49 as compared to 13). Figure 5 illustrates good graph fitting for all three instances. Generally, predictions were more accurate for a lower number of active S-boxes ($\lessapprox 10$) and a larger number of active S-boxes ($\gtrapprox 40$). As prediction errors are more significant for a smaller number of rounds that have a lower number of active S-boxes, we conclude that both the MAE and RMSE loss functions respectively led to slightly better prediction performance as compared to the quantile loss function based on the quantitative results in Table 6 and the curve fitting observed in Figure 5. However, as there are only minor differences in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

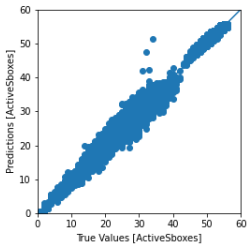
performance, we still used all three loss functions in Phase 2-1.

Loss Function	RMSE	R^2
MAE	0.67	0.994
RMSE	0.71	0.995
Quantile(90%)	0.96	0.992

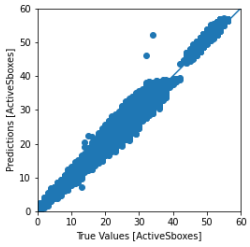
TABLE 6: Predicted number of active S-boxes based on loss functions (16-branch GFS)



(a) Prediction error for MAE loss function



(b) Prediction error for RMSE loss function



(c) Prediction error for quantile loss function

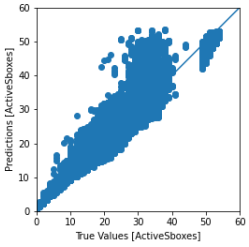
FIGURE 5: Comparison of loss functions for baseline experiment (16-branch GFS)

Experimental results in Table 7 show that the deep learning models are capable of predicting the number of active S-boxes for the unseen cipher, TWINE based on data from five other GFS ciphers. In other words, the model can generalize well to a cipher it has never seen before based on data from the ones that it already has. The neural network model with RMSE as its loss function had the lowest prediction errors (RMSE=1.62) and the highest R^2 score (0.87). The use of the quantile loss function led to the worst performance of the three.

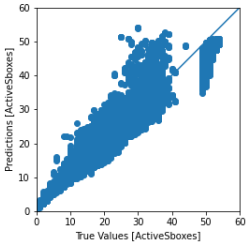
Overall, prediction performance faltered as compared to the baseline experiments. This was an expected outcome because predictions for TWINE were made based on what the deep learning models learned from other GFS ciphers. Prediction errors when using the MAE, RMSE, and quantile loss functions doubled respectively as compared to their baseline counterparts. From the cryptographic perspective, this is equivalent to an increase from ± 1 to ± 2 active S-boxes. Considering that a full-sized block cipher such as TWINE can have a large number of active S-boxes (50 or more even at 8 rounds), a deviation of two active S-boxes is small in comparison. Therefore, a cryptanalyst must account for a possible deviation of two active S-boxes when using the proposed approach in a practical attack scenario.

Loss Function	RMSE	R^2
MAE	1.71	0.86
RMSE	1.62	0.87
Quantile(90%)	2.31	0.83

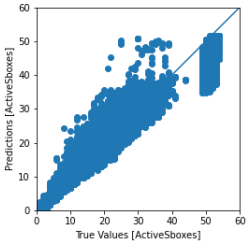
TABLE 7: Predicted number of active S-boxes based on loss functions (TWINE)



(a) Prediction error for MAE loss function



(b) Prediction error for RMSE loss function



(c) Prediction error for quantile loss function

FIGURE 6: Comparison of loss functions for generalization experiment (TWINE)

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3099802, IEEE Access

V. DISCUSSION AND FUTURE WORK

Our preliminary experiments on the 4-branch GFS showed that neural networks can be trained to predict the number of active S-boxes of a block cipher based on block cipher features that are common to more than just one block cipher. We also discovered that among these features, the permutation pattern plays the biggest role in reducing prediction error. However, we still recommend the inclusion of all other supporting features, such as truncated input and output differences, as they all contribute towards minimizing prediction error. Next, we showcased the feasibility of the proposed approach when being applied to actual lightweight GFS block ciphers such as TWINE. Apart from achieving even smaller prediction errors as compared to the preliminary experiments, the trained model was able to generalize well to block ciphers that it has yet seen. Based on our findings, we recommend the use of the following neural network architecture for future S-box prediction efforts, notably for 64-bit GFS ciphers:

- Input layer: 49 neurons (equivalent to the number of training features)
- Output layer: 1 neuron
- Hidden layer: 4 hidden layers, each with 512 neurons
- Loss function: RMSE
- Regularizer: regularizers.l2(0.001) (L2 regularization)
- Optimizer: Adam with 0.001 learning rate
- Activation Function: ELU
- Epochs: 200
- Batch size: 32

The capability of the deep learning models to generalize to the same block cipher structure they have been trained on is one of the main strengths of the proposed method. It is more flexible than prior cipher-specific approaches such as SAT or MILP because there is also no need to retrain the model if other block ciphers with a similar structure need to be analyzed. Even if retraining is required, it only takes approximately 40 and 70 minutes to train the model with 400000 training samples for the 4-branch and 16-branch ciphers respectively. In addition, the predictions are nearly instantaneous. Thus, the *active* phase of a cryptanalytic attack can be made more efficient as the bulk of the computational work has been moved to pre-processing (the training phase). This can be seen as a type of cryptanalytic time/memory/data trade-off [36], [37]. The proposed approach is also a trade-off between efficiency and exactness (predictions rather than actual values).

To illustrate this trade-off in practice, we compared the performance of a branch-and-bound search [8] and the proposed deep learning model when labeling the number of active S-boxes for 10000 randomly selected truncated difference pairs for TWINE. The amount of time required to label these pairs with respect to the number of rounds is depicted in Figure 7. The time required by the branch-and-bound search increased from approximately 2 milliseconds for 1 round to 9.7 hours for 8 rounds whereas the deep learning model had a constant time of around 1.2 seconds. To calculate the number of active S-boxes for a given truncated plaintext-ciphertext pair, the branch-and-bound search algorithm needs to traverse every possible branch until a match is discovered. The number of branches increases exponentially with the number of rounds, leading to an exponential decline in efficiency. This exponential increase in computational complexity also occurs in other differential search methods such as MILP [11]. In contrast, the prediction speed for the deep learning model is independent of the number of rounds. However, as compared to the branch-and-bound search which is an exact method, the deep learning model is bound to have prediction errors. For example, when performing predictions for up to 8 rounds of TWINE, the best-performing model achieved an R^2 score of 0.87. This can be estimated as a 13% degradation in terms of error in exchange for an efficiency gain of approximately 31000%.

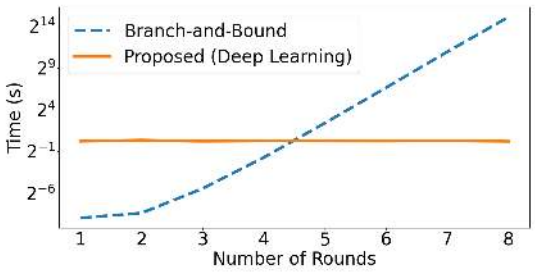


FIGURE 7: Time required to label 10000 randomly selected truncated differential pairs

Performing cryptanalysis usually requires niche expertise which includes a strong understanding of the underlying structure of a block cipher and the intricacies of the cryptanalytic method itself. The proposed method alleviates some of the technical expertise required to analyze block ciphers, making it a more data-driven approach. To evaluate a block cipher (with the same underlying structure) using the trained model, one would just need to use the block cipher itself to generate plaintext-ciphertext pairs for testing. There is no need to represent the block cipher as constraints to be solved by a SAT solver or to modify a branch-and-bound algorithm to suit the cipher's structure. The trained model can potentially be utilized in a multitude of ways such as filtering differential pairs for attacks, block cipher security evaluation for fast design prototyping, and security benchmarking. These potential applications will be explored further in future work.

However, the proposed method is not without its drawbacks. Firstly, as a data-driven approach, it relies on having a sufficiently large dataset which may take a long time to generate. For example, generating data samples for 8 rounds of a 16-branch GFS took two days to complete. The increase in search time is exponential with respect to the number

of rounds and block size. This can be alleviated by identifying suitable bounding criteria for the branch-and-bound algorithm that can ensure that examples are still randomly sampled without having to exhaustively explore each branch. Another drawback is that prediction performance falters when labeling data generated from more rounds than the model has been trained for. This is due to the fact that there are many instances where the number of active S-boxes is out of the range used in training. One way to overcome this issue is by labeling the dataset with a new metric that scales the number of active S-boxes with respect to the number of block cipher rounds. Last but not least, the current work has only delved into GFS ciphers. More research is still required to identify suitable block cipher features that can be used for SPN or ARX ciphers, which may need to take into account varying S-box sizes.

VI. CONCLUSION

In this paper, we proposed a deep learning approach to block cipher security analysis. Specifically, we train fully connected, deep neural network models to predict the number of active S-boxes, trading off exactness for efficiency. The active S-box prediction is framed as a regression task, whereby the fully connected, deep neural networks are trained using common block cipher features such as the number of rounds and permutation pattern, and differential cryptanalysis-related features such as truncated input and output differences. We investigate the feasibility of the proposed approach and the effect of block cipher features and their corresponding data representations on prediction error by applying it to smaller-scale (4-branch) GFS ciphers. The best performing models achieved low RMSE scores of under 1.96, which translates to prediction errors of approximately ± 2 S-boxes. High R^2 values ranging between 0.88-0.89 also indicate that there is a strong relationship between predicted and actual values. In terms of the features, the permutation pattern has the biggest impact on prediction performance as its removal leads to the largest increase in prediction errors (88.8%) as compared to the truncated input (23%) and output differences (15.8%). We also found that splitting these features into multiple smaller features reduces error. When applied to full-scale (16-branch) lightweight GFS block ciphers, the neural network models achieved low RMSE scores of under 0.96, and R^2 values ranging between 0.992-0.994. The deep learning models were also able to generalize to a cipher that they have not seen before, specifically the lightweight block cipher TWINE. The best performing model was able to predict the number of active S-boxes for TWINE with an RMSE of 1.62 and R^2 of 0.87. Overall, our findings showcase the feasibility of the proposed approach which can be used for rapid security assessment (which in turn contributes towards faster block cipher prototyping), security benchmarking, and also to aid cryptanalytic efforts.

REFERENCES

[1] G. Bansod, N. Raval, and N. Pisharoty, "Implementation of a New Lightweight Encryption Design for Embedded Security," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 142–151, jan 2015.

[2] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems," *Journal of Cryptology*, vol. 4, no. 1, pp. 3–72, jan 1991.

[3] F. Yu, X. Gong, H. Li, and S. Wang, "Differential cryptanalysis of image cipher using block-based scrambling and image filtering," *Information Sciences*, vol. 554, pp. 145–156, apr 2021.

[4] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "PRESENT: An Ultra-Lightweight Block Cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007*. Springer Berlin Heidelberg, pp. 450–466.

[5] W. Wu and L. Zhang, "LBlock: A Lightweight Block Cipher," in *Applied Cryptography and Network Security*. Springer Berlin Heidelberg, 2011, pp. 327–344.

[6] T. Suzaki, K. Minematsu, S. Morioka, and E. Kobayashi, "TWINE : A Lightweight Block Cipher for Multiple Platforms," in *Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2013, pp. 339–354.

[7] M. Sajadieh, A. Mirzaei, H. Mala, and V. Rijmen, "A new counting method to bound the number of active S-boxes in Rijndael and 3D," *Designs, Codes and Cryptography*, vol. 83, no. 2, pp. 327–343, may 2016.

[8] J. Chen, J. Teh, Z. Liu, C. Su, A. Samsudin, and Y. Xiang, "Towards Accurate Statistical Analysis of Security Margins: New Searching Strategies for Differential Attacks," *IEEE Transactions on Computers*, vol. 66, no. 10, pp. 1763–1777, oct 2017.

[9] W.-Z. Yeoh, J. S. Teh, and J. Chen, "Automated Search for Block Cipher Differentials: A GPU-Accelerated Branch-and-Bound Algorithm," in *Information Security and Privacy*. Springer International Publishing, 2020, pp. 160–179.

[10] N. Mouha and B. Preneel, "Towards Finding Optimal Differential Characteristics for ARX: Application to Salsa20," Cryptology ePrint Archive, Report 2013/328, Nov. 2013, https://eprint.iacr.org/2013/328. [Online]. Available: https://eprint.iacr.org/2013/328/20131113:001621

[11] S. Sun, L. Hu, P. Wang, K. Qiao, X. Ma, and L. Song, "Automatic Security Evaluation and (Related-key) Differential Characteristic Search: Application to SIMON, PRESENT, LBlock, DES(L) and Other Bit-Oriented Block Ciphers," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014, pp. 158–178.

[12] R. L. Rivest, "Cryptography and machine learning," in *Advances in Cryptology — ASIACRYPT '91*. Springer Berlin Heidelberg, 1993, pp. 427–439.

[13] A. Gohr, "Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning," in *Advances in Cryptology – CRYPTO 2019*. Springer International Publishing, 2019, pp. 150–179.

[14] K. V. Pradeepthi, V. Tiwari, and A. Saxena, "Machine Learning Approach for Analysing Encrypted Data," in *2018 Tenth International Conference on Advanced Computing (ICoAC)*. IEEE, dec 2018.

[15] S. Pamidiparthi and S. Velampalli, "Cryptographic Algorithm Identification Using Deep Learning Techniques," in *Evolution in Computational Intelligence*. Springer Singapore, sep 2020, pp. 785–793.

[16] V. Tiwari, K. V. Pradeepthi, and A. Saxena, *Identification of Cryptographic Algorithms Using Clustering Techniques*, K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi, and M. R. Murty, Eds. Springer Singapore, 2020.

[17] W. Zhang, Y. Zhao, and S. Fan, "Cryptosystem Identification Scheme Based on ASCII Code Statistics," *Security and Communication Networks*, vol. 2020, pp. 1–10, dec 2020.

[18] A. Gomez, S. Huang, I. Zhang, B. Li, M. Osama, and L. Kaiser, "Unsupervised Cipher Cracking Using Discrete GANs," in *International Conference on Learning Representations*, 2018.

[19] G. Mishra, S. V. S. S. N. V. G. K. Murthy, and S. K. Pal, "Neural Network Based Analysis of Lightweight Block Cipher PRESENT," in *Harmony Search and Nature Inspired Optimization Algorithms*. Springer Singapore, aug 2018, pp. 969–978.

[20] A. Jain and G. Mishra, "Analysis of Lightweight Block Cipher FeW on the Basis of Neural Network," in *Harmony Search and Nature Inspired Optimization Algorithms*. Springer Singapore, aug 2018, pp. 1041–1047.

[21] Y. Xiao, Q. Hao, and D. D. Yao, "Neural Cryptanalysis: Metrics, Methodology, and Applications in CPS Ciphers," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, nov 2019.

[22] A. Mundra, S. Mundra, J. S. Srivastava, and P. Gupta, "Optimized deep neural network for cryptanalysis of DES," *Journal of Intelligent & Fuzzy Systems*, vol. 38, pp. 5921–5931, 2020.

[23] A. Perov, "Using Machine Learning Technologies for Carrying out Statistical Analysis of Block Ciphers," in *2019 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. IEEE, oct 2019.

[24] N. D. Truong, J. Y. Haw, S. M. Assad, P. K. Lam, and O. Kavehei, "Machine Learning Cryptanalysis of a Quantum Random Number Generator," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 403–414, feb 2019.

[25] S. Saha, D. Jap, S. Patranabis, D. Mukhopadhyay, S. Bhasin, and P. Dasgupta, "Automatic Characterization of Exploitable Faults: A Machine Learning Approach," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 4, pp. 954–968, apr 2019.

[26] A. G. Bafghi, R. Safabakhsh, and B. Sadeghiyan, "Finding the differential characteristics of block ciphers with neural networks," *Information Sciences*, vol. 178, no. 15, pp. 3118–3132, aug 2008.

[27] A. Baksi, J. Breier, Y. Chen, and X. Dong, "Machine Learning Assisted Differential Distinguishers For Lightweight Ciphers (Extended Version)," Cryptology ePrint Archive, Report 2020/571, Dec. 2020, https://eprint.iacr.org/2020/571. [Online]. Available: https://eprint.iacr.org/2020/571/20201202:014352

[28] B. Hou, Y. Li, H. Zhao, and B. Wu, "Linear Attack on Round-Reduced DES Using Deep Learning," in *Computer Security – ESORICS 2020*. Springer International Publishing, 2020, pp. 131–145.

[29] T. R. Lee, J. S. Teh, J. L. S. Yan, N. Jamil, and W.-Z. Yeoh, "A Machine Learning Approach to Predicting Block Cipher Security," in *Cryptology and Information Security Conference*. Universiti Putra Malaysia, 2020.

[30] J. So, "Deep Learning-Based Cryptanalysis of Lightweight Block Ciphers," *Security and Communication Networks*, vol. 2020, pp. 1–11, jul 2020.

[31] J. Zou, Y. Han, and S.-S. So, "Overview of Artificial Neural Networks," in *Methods in Molecular Biology™*. Humana Press, 2008, pp. 14–22.

[32] R. Z. Bharath Ramsundar, *TensorFlow for Deep Learning*. O'Reilly UK Ltd., Apr. 2018.

[33] J. Chen, A. Miyaji, C. Su, and J. S. Teh, "Accurate Estimation of the Full Differential Distribution for General Feistel Structures," in *Information Security and Cryptology*. Springer International Publishing, 2016, pp. 108–124.

[34] Y. Zheng, T. Matsumoto, and H. Imai, "On the Construction of Block Ciphers Provably Secure and Not Relying on Any Unproved Hypotheses," in *Advances in Cryptology — CRYPTO' 89 Proceedings*. Springer New York, pp. 461–480.

[35] T. Suzaki and K. Minematsu, "Improving the Generalized Feistel," in *Fast Software Encryption*. Springer Berlin Heidelberg, 2010, pp. 19–39.

[36] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE Transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, jul 1980.

[37] A. Biryukov and A. Shamir, "Cryptanalytic time/memory/data tradeoffs for stream ciphers," in *Advances in Cryptology — ASIACRYPT 2000*. Springer Berlin Heidelberg, 2000, pp. 1–13.

MOHAMED FADL IDRIS received the B.Eng. degree (Hons.) majoring in Information Technology from Sudan Open University, Sudan in 2016, then received his M.Sc. in Information Technology from Omdurman Islamic University in 2018. He is currently pursuing his Ph.D. at the School of Computer Sciences, Universiti Sains Malaysia in the area of machine learning, symmetric cryptography, and cryptanalysis.

JE SEN TEH received the B.Eng. degree (Hons.) majoring in Electronics from Multimedia University, Malaysia in 2011, then received his M.Sc. and Ph.D. in Computer Science from Universiti Sains Malaysia in 2013 and 2017 respectively. He is currently working as a Senior Lecturer in Universiti Sains Malaysia under the School of Computer Sciences. His research interests include symmetric cryptography, cryptanalysis and chaos theory.

JASY LIEW SUET YAN is a Senior Lecturer at the School of Computer Sciences, Universiti Sains Malaysia specializing in sentiment analysis, natural language processing, and machine learning. Her broader research interests include text mining, computational linguistics, and deep learning. She completed her Masters in Information Management and Ph.D. in Information Science and Technology from the School of Information Studies, Syracuse University, USA.

WEI-ZHU YEOH received the B.Sc. (Hons) degree in Computer Science from Coventry University (a joint program at INTI International College Penang) in 2018, then he received the M.Sc. degree in Computer Science from Universiti Sains Malaysia (USM) in 2020. He is currently pursuing his Ph.D. at CISPA Helmholtz Center for Information Security, Saarland University starting 2021. His research interests include symmetric cryptography, machine learning application in cryptography, parallel computing, and mobile computing.