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

Mohamed Fadl Idris, Je Sen Teh*, Jasy Liew Suet Yan, Wei-Zhu Yeoh

***Abstract*—Resistance against differential cryptanalysis is commonly assessed by counting the number of active substitution boxes (s-boxes) using search algorithms or mathematical solvers that incur high computational costs. In this paper, we propose an alternative approach using deep neural networks to predict the number of active s-boxes, trading 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) generalised 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 accuracy. We then extend the idea to 64-bit GFS ciphers by training deep learning models using data from five ciphers. These deep learning models were then used 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 a myriad of other cryptographic algorithms and protocols. More recently, lightweight block ciphers have been gaining popularity to provide data security to resource-constrained devices such as RFID tags or smart Internet-of-Things (IoT) devices. Depending on the underlying structure of this function, modern block ciphers can be classified into different categories such as the generalised Feistel structure (GFS), addition-XOR-rotate (ARX) and substitution-permutation network (SPN). 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. Keys for each encryption round, known as round keys, are derived from a master key using a key scheduling algorithm. A block cipher is considered to be secure enough for practical applications if it is 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 a ciphertext differences (output differences) [1]. Even today, resistance against differential cryptanalysis is considered one of the de facto requirements for new block cipher designs. 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 [2], [3], [4]. 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 holds for an s-box. p_{sbox} is obtained by deriving a differential distribution table for a particular s-box. Resistance is estimated using the worst-case probability, which for optimal 4 and 8-bit s-boxes is 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 with minimal active s-boxes. Identification of the best differential trail is a computationally intensive task which can be performed using searching algorithms [5], [6] or mathematical solvers [7], [8].

Recently, concepts from differential cryptanalysis have been used for an entirely different purpose: to train machine learning algorithms for cryptanalysis applications. [9] introduced the first successful application of machine learning (or more specifically, deep learning) in the context of classical crypt-

*Corresponding Author
All authors are both with the School of Computer Sciences, Universiti Sains Malaysia.

analysis. A machine learning-based differential distinguisher was developed and used to cryptanalyze a round-reduced Speck32/64. His findings indicate that the machine learning distinguishers were superior to conventional differential distinguishers. [10] expanded on 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. These recent findings support the fact that machine learning has huge untapped potential when used alongside classical cryptanalysis concepts. However, both of these approaches are specific to individual ciphers and would require the machine learning models to be retrained when a different cipher is to be analyzed.

A. Related Work

The application 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 that have been encrypted by popular block ciphers such as AES, 3DES, Blowfish and Camellia [11], [12], [13], [14]. 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, [15] utilized unsupervised learning with neural networks to attack classical ciphers such as the Vigenere and Shift ciphers. [16] investigated whether neural networks can be used to predict plaintext bits of the block cipher PRESENT based on data obtained from any particular round. Their proposed approach was not able to predict bits at any of the 64-bit positions with a probability significantly greater than half. The same approach was used in [17] against the FeW cipher with limited success. [18] described a black-box security evaluation approach to measure the strength of proprietary ciphers without knowledge of the encryption algorithms themselves. They quantify the strength of a cipher by measuring how difficult it is for a neural network to mimic the cipher algorithm. Their results show that the security strength of Hitag2 (a cipher used for keyless entries in modern cars) is weaker than a 3-round DES. Cipher mimicking has also been explored by [19]. The researchers optimized a deep neural network model to decrypt 64-bit DES ciphertexts without knowledge of the secret key, achieving an accuracy of up to 90%.

[20] demonstrated that using machine-learning techniques, it is possible to distinguish the ciphertexts of round-reduced ciphers from random sequences. This can be achieved by with a smaller sample size (2^{12} bits) as compared to conventional statistical methods. In 2019, [9] introduced the first successful application of machine learning from the perspective of conventional cryptanalysis. 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. [10] 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 [21]. [22] 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, [23] 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 is restricted to text-based keys.

B. Contribution

In this paper, we train (deep) neural network models to predict the number of active s-boxes for lightweight block ciphers based on 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 are used as labels. We generate training and testing data using a modified branch-and-bound search from [5]. Unlike past 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 architectures for the prediction task in a practical amount of time. In addition, we explored the impact of each block cipher feature and its representation on prediction accuracy. We found that fully connected neural network architectures 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. Results show that the permutation pattern has the biggest impact on prediction accuracy 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 [4] was used as the target cipher. The neural network model was trained using data generated from five generic GFS ciphers, each with different permutation patterns and limited to eight rounds. Then, the

trained model was used to label data samples taken from TWINE (also limited to eight rounds). 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 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 rapid assessment of block cipher security to the filtering differential pairs for cryptanalytic attacks.

C. Outline

The rest of this paper is structured as follows: Background information on neural networks, truncated differentials and GFS are provided in Section II. The experimental setup and results are detailed in Sections III and IV for the small and full-scale lightweight GFS ciphers respectively. Section V discusses the important findings and potential applications of the proposed work before the paper is concluded in Section VI.

II. Preliminaries

A. Neural Networks

A neural network composes of connected processing units called neurons. As its name implies, it is a network of neurons divided into multiple layers - an input layer, output layer, and one or more hidden layers. Neural networks have been used for various tasks, from speech recognition to real-time electrocardiogram monitoring [24]. Each artificial neuron can receive multiple inputs and produces a single output that can be mapped to other neurons. Figure 1 illustrates the structure of a fully connected neural network.

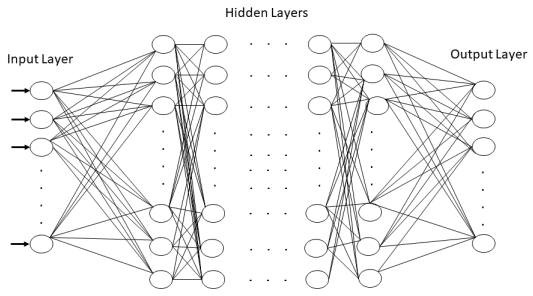


Fig. 1: Neural Network Example

In general, machine learning algorithms either minimize or maximize a function known as an *objective function*. The functions that machine learning algorithms try to minimize are known as *loss functions*. A loss function can be used as a measure of how well a neural network model can predict the expected outcome. Figure 2 illustrates a typical loss function where the red dots represent the actual values, and the red line represents the predicted values. Some of the loss functions that are used to minimize prediction errors 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. RMSE is also used as a performance measurement metric.
- **Mean absolute error (MAE)** calculates the average of the absolute values of prediction errors for all instances in a dataset. MAE is also used as a performance measurement metric.
- 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.

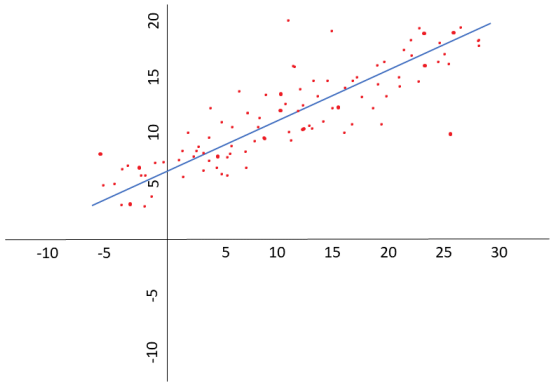


Fig. 2: Loss Functions

The proposed s-box prediction task is framed as a regression problem. Regression estimates the relationship between a dependent variable and one or more independent variables. RMSE and R-squared (R^2) are two performance evaluation metrics selected for the proposed work. As previously described, 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. 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.

Model overfitting is a machine learning problem that occurs when a model does not generalize well from training to unseen (or test) data. Several methods exist to prevent overfitting such as cross-validation, having more training data, feature selection, early stopping, and regularization. In the proposed work, 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.

B. Truncated Differential Cryptanalysis

Differential cryptanalysis relies on identifying an r-round differential trail, $\Delta X \xrightarrow{r} \Delta Y$ with high probability of occurring. 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 [25], 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 differences if 4-bit ($t = 4$) s-boxes. 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 3.

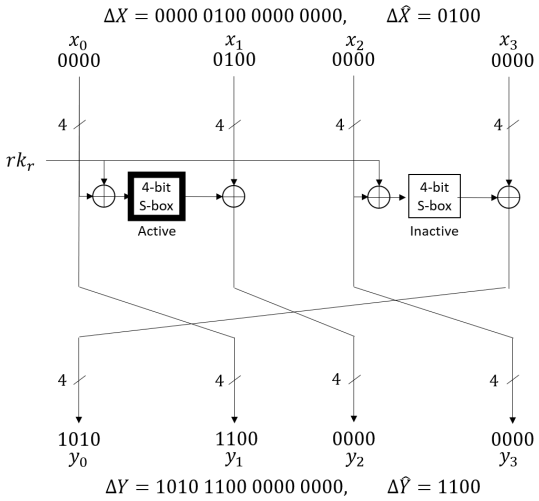


Fig. 3: 4-branch GFS (1 round)

C. Generalized Feistel Structure and TWINE

One of the common structures for a block cipher is GFS, a generalization of the Feistel structure first used by the block cipher Lucifer (the precursor to DES). A GFS divides an input into k sub-blocks, where $t > 2$. The size of these sub-blocks is usually dependent on the s-box size. The Type-II GFS [26], [27] 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 3 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. An s-box is considered *active* 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 3 is $P = \{1, 2, 3, 0\}$. The structure of TWINE 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 is also based on the Type-II GFS with 16 4-bit sub-blocks. The size of the sub-blocks corresponds to the size of the s-box used in TWINE. TWINE will be used as the target cipher for our generalization experiments.

III. Active S-box Prediction for 4-branch GFS

The overall goal of the proposed work is to train neural network models to predict the number of active s-boxes of a block cipher based on features such as truncated differences, the number of rounds and permutation pattern. This is a supervised learning problem which we frame as a regression task. This section describes the first set of experiments performed on 4-branch GFS ciphers as a proof-of-concept. Our experiments are divided into two main phases: In Phase 1, we identify the best performing neural network architecture to become our baseline model. In this phase, we also determine if splitting certain features into multiple smaller ones can lead to improved prediction accuracy. In Phase 2, we examine the effect of removing certain features from the training data on prediction accuracy. Prediction accuracy is measured using RMSE and R^2. All experiments are implemented using Tensorflow 2.3.1 on an Intel Core i5 Processor, 8GB RAM using Python and Jupyter Notebook.

A. Phase 1 - Baseline Experiments (4-branch GFS)

1) Experimental Setup: Using the smaller scale 4-branch GFS for proof-of-concept experiments allows us to generate a large amount of training/testing data within a shorter amount of time. In addition, data for all 24 possible permutation patterns (each of which can be viewed as individual 4-branch GFS ciphers) can be generated. We generate a dataset of 500000 samples using a modified branch-and-bound differential search algorithm[1]. The dataset consists of randomly sampled differential data from round 1 until round 12 of the 24 GFS ciphers. The format of the data samples is depicted in Table I.

The first experiment involves training a baseline model using all the available block cipher features such as the truncated input difference $\Delta \hat{X}$, truncated output difference $\Delta \hat{Y}$, number

[1]The search algorithm is available at https://github.com/jesenteh/16b-gfs-as-search.

$\Delta\hat{X}$	$\Delta\hat{Y}$	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 I: Examples of data samples for the 4-branch GFS

of rounds, r and permutation pattern, P. Each sample in the dataset is labelled with the number of active s-boxes, AS. A train-test split with a ratio of 80:20 is used for the experiments (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 I. However, we later found that the trained models perform better when the truncated difference and permutation features were split into four separate variables (four features) each as shown in Table II. In the following subsection, we only report results from experiments that use this alternative representation.

$\Delta\hat{X}$	$\Delta\hat{Y}$	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 II: Examples of data samples for the 4-branch GFS with alternate representation

Due to the strong confusion and diffusion capabilities of a block cipher, every ciphertext bit is strongly influenced by all plaintext and key bits. Thus, we have chosen fully connected neural networks for the proposed work as full connectivity can better represent the relationships between these bits. An example of a fully connected neural network is shown in Figure 1. We then perform hyperparameter tuning to identify the best neural network architecture, loss function, optimizer, number of epochs and batch size for the prediction task. Based on our experiments, we selected a neural network architecture with 12 neurons for the input layer (equivalent to the number of input features), four hidden layers with 64 neurons each, and an output layer with one neuron to represent the predicted number of active s-boxes. The rest of the hyperparameters used in our experiments are summarized below:

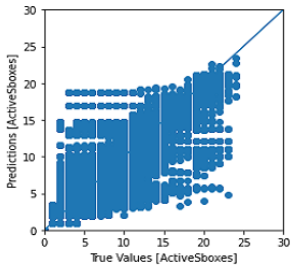
- Optimizer: Adam with 0.001 learning rate
- Activation Function: Exponential linear unit (ELU)
- Epochs: 100 epochs
- Batch Size: 32

2) Results: The performance of proposed deep learning model for three loss functions (MAE, RMSE and quantile) is compared in Table III. Overall, the models can predict the number of active s-boxes with an RMSE of just under 1.96. 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.

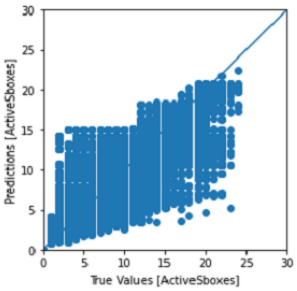
From the perspective of differential cryptanalysis, two 4-bit active s-boxes would incur a probability penalty of $2^{-2\times AS} = 2^{-4}$. This probability penalty is more significant for a smaller number of rounds as the number of active s-boxes is generally lower. As the quantitative results in Table III indicate that all three models perform similarly, deciding which loss function leads to the best result will depend on how accurately it can predict smaller active s-box numbers. This is depicted in Figure 4 which illustrates that the quantile loss function leads to slightly better curve fitting as compared to the rest, notably when the number of active s-boxes is smaller. Therefore, the neural network model with the quantile loss function is chosen as the baseline model.

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

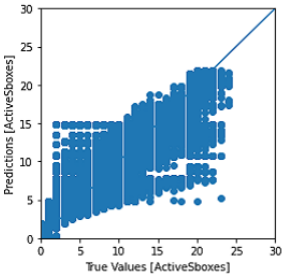
TABLE III: Predicted number of active s-boxes based on loss functions (4-branch GFS)



(a) Prediction accuracy for MAE loss function



(b) Prediction accuracy for RMSE loss function



(c) Prediction accuracy for quantile loss function

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

B. Phase 2 - Feature Refinement (4-branch GFS)

1) Experimental Setup: In Phase 2, we investigate the effect of specific block cipher features on prediction accuracy. This is performed by removing only one of the features (truncated output difference, truncated input difference or permutation) then using the remaining ones for training. The same fully connected neural network architecture as the baseline model is used, which has 12 neurons in the input layer, followed by four hidden layers of 64 neurons, and finally an output layer with one neuron. The same dataset and hyperparameters as the baseline model are also used for all experiments.

2) Results: The experimental results are shown in Table IV. Generally, removal of either the truncated input or output difference has a minor negative effect on prediction accuracy. In contrast, removal of the permutation pattern has a significant impact on the model's performance. Experimental evidence from Table IV suggests that the input difference, $\Delta\hat{X}$ plays a slightly bigger role than the output difference, $\Delta\hat{Y}$. Removal of $\Delta\hat{X}$ leads to a 23% increase in prediction errors as compared to the baseline whereas there is an 15.8% increase when $\Delta\hat{X}$ is removed. In both cases, R^2 values only reduced by an average of 6.3% as compared to the baseline, implying that there is still a strong relationship between actual and predicted values.

Intuitively, removal of the permutation pattern should have an adverse effect on prediction accuracy because a neural network would need to learn how to differentiate between 24 different block ciphers based entirely on the input and output truncated differences. Due to the strong diffusive property of block ciphers, this would be a difficult task. Our experimental results confirms this notion. In terms of prediction errors, there is a significant increase (88.8%) in prediction errors (RMSE) as compared to the baseline model whereas the R^2 score dropped by 31.8%. This shows that the deep learning model relies heavily on the permutation pattern to make accurate predictions of the number of active s-boxes.

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 IV: Predicted number of active s-boxes when removing specific features

IV. Active S-box Prediction for 16-branch GFS

Our preliminary proof-of-concept experiments on smaller-scale GFS ciphers indicate that a neural network can learn the relationship between block cipher features and the number of active s-boxes. Next, we extend the same idea to 64-bit lightweight block ciphers, notably 16-branch GFS block ciphers such as TWINE. We also perform experiments to determine if a deep learning model trained using data from several ciphers can generalize to a cipher that it has not *seen* before. This showcases the improved flexibility of the proposed method as compared to existing machine learning-based cryptanalysis methods that are cipher-specific [9], [10]. Our experiments on 16-branch GFS ciphers are divided into two main phases: In Phase 1, we identify the best performing neural network architecture to become our baseline model. Phase 2 is the generalization experiment where we use the trained model to label a dataset generated from TWINE. Prediction accuracy is 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) is used here as well.

A. Phase 1 - Baseline Experiment (16-branch GFS)

1) Experimental Setup: As it is not practical to enumerate every possible permutation pattern or block cipher variant for 16-branch GFS, we select six for our experiments. The first is TWINE itself which is our target cipher whereas the other five are based on permutation patterns with the best cryptographic properties based on findings in [27]. 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 V.

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 V: 16-branch permutation patterns

The dataset is generated using the same modified branch-and-bound search. 100000 data samples are generated for each cipher, consisting of 12500 randomly selected samples from round 1 to round 8. Apart from being able to generate data samples within a practical amount of time, we selected 8 rounds as it is the minimum number of rounds required to achieve for full diffusion [27]. We limit the number of output differences per input difference to four to ensure that there is a larger variety of input differences in the dataset. The format of the data samples is similar to the one depicted in Table I.

The first experiment involves training a baseline model using all the available block cipher features ($\Delta\hat{X}$, $\Delta\hat{Y}$, r and P). Each sample in the dataset is labelled with the number of active s-boxes, AS. The data samples for Phase 1 are taken from all ciphers apart from TWINE. A train-test split with a ratio of 80:20 used for the experiments (400000 training samples, 100000 test samples). In terms of feature representation, we split the truncated input and output difference, and permutation pattern into 16 separate features. Thus, 49 features are used for training. The neural network used for the experiment still has six layers, one input layer, four hidden layers, and one output layer. To accommodate the larger number of input features, the input layer of the neural network is increased to 49. We also increased the number of neurons in the hidden layers to 512. The hidden layer still has only one neuron. The remaining hyperparameter values for the experiment are as follows:

- Optimizer: Adam with 0.001 learning rate
- Activation Function: ELU

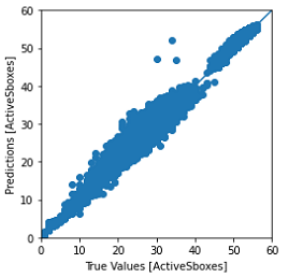
- Epochs : 200 epochs
- Batch Size: 32

2) Results: The performance of proposed deep learning model for three loss functions (MAE, RMSE and quantile) is compared in Table VI. Generally, the proposed model has low prediction errors when using all three loss functions, achieving RMSE values of under 0.96. From a cryptanalyst's perspective, results show that the proposed models can be used to predict the number of active s-boxes with high accuracy (± 1 active s-box) 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 predicted and real values. We note that there is a significant improvement in prediction performance in the 16-branch GFS as compared to the 4-branch GFS which could be due to the larger number of training features that has allowed the neural network to converge to the global optima.

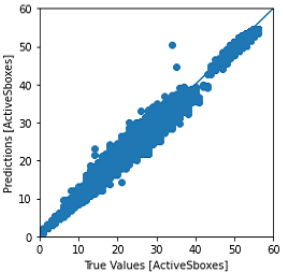
Figure 5 illustrates good graph fitting for all three instances. Generally, predictions are more accurate for a lower number of active s-boxes ($\lessapprox 10$) and a larger number of active s-boxes ($\gtrapprox 40$). As discussed in Section III-A2, prediction errors are more significant for a smaller number of rounds which have a lower number of active s-boxes. Therefore, we conclude that both MAE and RMSE lead to slightly better prediction performance as compared to RMSE based on both the quantitative results in Table VI and the graph fitting results in Figure 5. As there are only minor differences in performance, we will use all three for comparisons in the next phase.

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

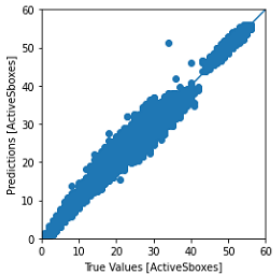
TABLE VI: Predicted number of active s-boxes based on loss functions (16-branch GFS)



(a) Prediction accuracy for MAE loss function



(b) Prediction accuracy for RMSE loss function



(c) Prediction accuracy for quantile loss function

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

B. Phase 2 - Generalization Experiment (16-branch GFS)

1) Experimental Setup: In this phase, we investigate if the proposed model is capable of labelling *unseen* data or in other words, generalize to a block cipher it has never encountered before. We use TWINE as our target block cipher. The 100000 data samples generated from round 1 to round 8 of TWINE is used as the testing data in this phase. The 500000 data samples from the other five ciphers are used for training. The same fully connected neural network architecture and hyperparameters as the baseline model is used for these experiments. However, rather than just testing one loss function, we will still investigate the use of all three (MAE, RMSE and quantile). In addition, we use the L2 regularizer (regularizers.l2(0.001) in TensorFlow) to overcome overfitting problems for better prediction performance.

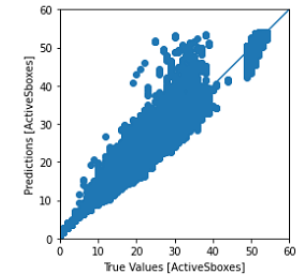
2) Results: Experimental results in Table VII show that the deep learning models are capable of predicting the number of active s-boxes for TWINE based on data obtained from five other ciphers. In other words, the model can generalize well to a cipher it has never seen before based on data from the ones that it already has. This has a wide array of potential applications that will be further discussed in Section V. The neural network model that uses RMSE as its loss function has the lowest prediction errors (RMSE=1.62). Its R^2 score is also the highest (0.87), indicating that there is a strong relationship between the predicted and actual number of active s-boxes. When it comes to graph fitting, the use of the MAE loss function slightly outperforms RMSE (as shown in Figure 6). Generally, the use of the quantile loss function leads to the worst performance of the three.

As expected, prediction performance has declined when compared to the baseline experiments. Prediction errors when using the MAE, RMSE and quantile loss functions have increased by 155%, 128% and 141% 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), an deviation in one active s-box in

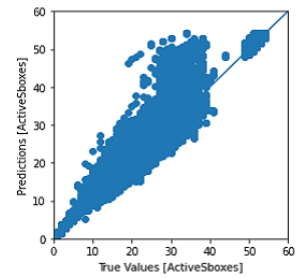
small in comparison.

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

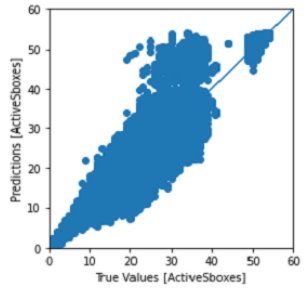
TABLE VII: Predicted number of active s-boxes based on loss functions (TWINE)



(a) Prediction accuracy for MAE loss function



(b) Prediction accuracy for RMSE loss function



(c) Prediction accuracy for quantile loss function

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

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 improving prediction accuracy. 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 errors. 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 to see. 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: MAE
- 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 other *unseen* block ciphers is one of the main strengths of the proposed method. It is more flexible than prior cipher-specific approaches because there is also no need to retrain the model if other similar block ciphers needs to be analyzed. 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 a cryptanalytic time/memory/data trade-off [28], [29]. The proposed approach is also a trade-off between efficiency and exactness (predictions rather than actual values).

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 requires alleviates some of the technical expertise required to analyze block ciphers, making it a more data-driven approach. A trained model can potentially be used 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 addressed in future work.

However, the proposed method is not without its drawbacks. Firstly, it is a data-driven approach that relies on having a sufficiently large dataset. This dataset 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. Another drawback is that the proposed deep learning models cannot predict the number of active s-boxes for more rounds than it has been trained for. One way to overcome this issue is by labelling the dataset with a new metric that scales the number of active s-boxes based on the number of rounds. This will again be left for future work. 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.

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 accuracy by applying it on 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 to prediction accuracy 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 improves accuracy. 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, the experimental results showcase the feasibility of the proposed approach which can be applied in security evaluation and cryptanalysis efforts.

Acknowledgment

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.

References

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

[2] 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.

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

[4] 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.

[5] 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.

[6] 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.

[7] 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

[8] 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.

[9] 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.

[10] 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

[11] 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.

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

[13] K. S. Raju, A. Govardhan, B. P. Rani, R. Sridevi, and M. R. Murty, Eds., *Proceedings of the Third International Conference on Computational Intelligence and Informatics*. Springer Singapore, 2020.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, apr 2021.

[25] 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.

[26] 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.

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

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

[29] 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.