Reducing the Cost of Machine Learning Differential Attacks Using Bit Selection and a Partial ML-Distinguisher

Amirhossein Ebrahimi[1], Francesco Regazzoni[2,3], and Paolo Palmieri[1]

[1] School of Computer Science & IT, University College Cork
{a.ebrahimimodhaddam,p.palmieri}@cs.ucc.ie
[2] University of Amsterdam, The Netherlands, f.regazzoni@uva.nl
[3] Università della Svizzera italiana, Switzerland, regazzoni@alari.ch

Abstract. In a differential cryptanalysis attack, the attacker tries to observe a block cipher's behavior under an input difference: if the system's resulting output differences show any non-random behavior, a differential distinguisher is obtained. While differential cryptanlysis has been known for several decades, Gohr was the first to propose in 2019 the use of machine learning (ML) to build a distinguisher.
In this paper, we present the first Partial Differential (PD) ML-distinguisher, and demonstrate its effectiveness on lightweight cipher SPECK32/64. As a PD-ML-distinguisher is based on a selection of bits rather than all bits in a block, we also study if different selections of bits have different impact in the accuracy of the distinguisher, and we find that to be the case. More importantly, we also establish that certain bits have reliably higher effectiveness than others, through a series of independent experiments on different datasets, and we propose an algorithm for assigning an effectiveness score to each bit in the block. By selecting the highest scoring bits, we are able to train a partial ML-distinguisher over 8-bits that is almost as accurate as an equivalent ML-distinguisher over the entire 32 bits (68.8% against 72%), for six rounds of SPECK32/64. The reduced input size implies a significant reduction in the complexity of achieving a distinguisher, and also leads to a reduction in the number of bits of possible subkeys to be guessed in a potential subsequent key recovery attack. These results may therefore open the way to the application of (partial) ML-based distinguishers to ciphers whose block size has so far been considered too large.

Keywords: Differential cryptanalysis · ML-based Differential cryptanalysis · Machine Learning · Partial ML-distinguisher.

1 Introduction

Block ciphers are cryptographic algorithms that provide confidentiality by encrypting data using a symmetric key. Block ciphers operate on fixed-length groups of bits, called blocks; rather than encrypting one bit at a time as in stream ciphers. Block ciphers are fundamental components in the design of many cryptographic protocols and are widely used to encrypt large amounts of data, either locally or over network communication. Recently, the cryptographic community has focused on the design of lightweight cryptographic (LWC) schemes, which are suitable for resource-constrained devices that are commonplace in settings such as the Internet of Things, healthcare, and sensor networks. The need for dedicated ciphers rises from the fact that the majority of current cryptographic algorithms were designed having desktop and server environments in mind, and because of this many of these algorithms are too computationally heavy to operate onto constrained devices. The US National Institute of Standards and Technology, who has a prominent role in the standardization of cryptogrphic algorithms recognized worldwide, has recently launched an initiative to solicit, evaluate, and standardize lightweight cryptographic algorithms [4], with the objective to achieve a set of standards for lightweight cryptographic algorithms by 2022.

Given their pervasive use, it is vital to evaluate the security of block ciphers, and especially those in the LWC domain, who have appeared more recently and have therefore been less studied than standard block ciphers such as the Advanced Encryption Standard (AES) [10]. In the research domain of cryptanalysis, which studies ciphers to find weaknesses and potential attacks, there are many generic and robust statistical techniques that can be used to attack algorithms and therefore help assess their security. The two most famous examples are *differential* [8] and *linear* [17] cryptanalysis. The main idea behind these attacks is to find a statistical pattern introduced by the cipher: this is achieved by looking at the ciphertexts produced by the algorithm, and by trying to distinguish between a random permutation and a block cipher. In a simple differential attack, which is usually a chosen plaintext attack, pairs of plaintext related by a constant difference (e.g. a logical XOR operation) are used. The attacker encrypts the plaintexts and computes the differences of the corresponding ciphertexts, in order to detect statistical patterns in their distribution. This pattern, whose statistical properties depend upon the nature of the S-boxes used for encryption, is called a *differential*. On the basis of the differential, the cipher can be distinguished from random, obtaining what is called a *distinguisher*.

Traditionally, the implementation of differential cryptanalysis techniques, however, requires a massive amount of data and memory, and the time complexity of finding a good distinguisher could be infeasible in most cases. Consequently, there is an active line of research aimed at automating these cryptanalysis methods. Until recently, the main focus was to transform the problem of finding a good distinguisher into an optimization problem [11, 18], which can then be solved more efficiently with optimization solvers like Gurobi [13]. Despite the more convenient approach for finding the best distinguisher, this process is still time-consuming, and the attacker still needs a good knowledge about the block cipher under attack.

For many years, it was believed that Machine Learning (ML) and cryptography could not work well together due to the random behavior of block ciphers and other cryptosystems [19, 1]. However, in a seminal paper in 2019 Gohr presented an ML-based cryptanalysis of the SPECK32/64 cipher that was better than previous attacks [12]. In that paper, it was illustrated that by using deep learning, a differential distinguisher could be achieved in an automated way and with less data complexity than other attacks, to the point where the cryptanalysis process can be implemented on a personal computer.

Using artificial intelligence (AI) techniques such as machine learning for the cryptanalysis of block ciphers can open many exciting opportunities. For instance, with the help of AI, it is possible to extend the known differential distinguishers for block ciphers [20]. In this paper, we focus on reducing the memory and computation costs of differential ML attacks by proposing the first partial differential ML-based distinguisher. In doing so, we also establish the first experimental differential evaluation of the role of each bit of the input and output of a cipher in its security.

1.1 Related work

In 2019, Aron Gohr introduced an 8-round differential distinguisher for SPECK32/64 cipher with the help of machine learning [12], and based on that, an 11-round attack was established, which was better than previous classic attacks. Gohr's central idea was using distinguishing attacks with the help of AI. He trained a neural classifier that can classify between a block cipher and a random permutation by looking at the output differences of the ciphertexts for a specific plaintext difference in SPECK32/64. He then compared this neural distinguisher with the traditional *all-in-one* differential distribution table of SPECK32/64, which was commutable due to cipher's small block size, and noticed ML-distinguishers could be a good model of it. Furthermore, he presented a method to find a good input difference for distinguishing attacks with the help of ML without any prior knowledge.

The comparison between *all-in-one* differential attack [2] and ML-based attack was possible because SPECK32/64 is a Markov cipher with a small block size. Following Gohr's intuition, a subsequent study was published by Baksi et al. [3] which used deep learning differential to train distinguishers for non-Markov ciphers, and on that basis could simulate non-Markov cipher's all-in-one differential distribution table. This was modeled successfully for ciphers with big state sizes such as Gimli. Moreover, the paper studied other architectures of deep learning networks, including Long Short-Term Memory (LSTM) and Multilayer Perceptron (MLP). Their results indicate that an MLP network with three hidden layers can be efficient enough to train a distinguisher.

Linear cryptanalysis has also been recently attempted using machine learning. Hou et al. applied machine learning on DES cipher to achieve a linear attack [14], using known plaintext and their corresponding ciphertexts. The results show that a neural network can recognize the XOR distribution of a linear expression in DES cipher. Other attacks such as integral have also been investigated in conjunction with machine learning [21].

Recent research in this direction is not limited to block ciphers: Liu et al. in [16] analyze the security of variants of Xoodyak hash mode against preimage attack utilizing deep learning. They trained a model for one round of permutation to predict the message of a hash function and discover that the accuracy is high. However, as the number of rounds is increased, the deep learning preimage attack diminishes in effectiveness.

Benamira et al. contributed a more in-depth analysis of the functioning of ML-based distinguishers, and focused in particular on what information they use [7]. Their results indicate that these machines not only use the differential distribution on ciphertext pairs, but the distinguisher depends on the penultimate or antepenultimate rounds. Based on these findings, they proposes a new pure cryptanalysis distinguisher with the same accuracy as Gohr's neural distinguisher.

An important limitation of current attacks relates to their complexity. To attack $n+1$ rounds of a block cipher using n-round of Gohr's neural distinguisher, we need to guess all the possible last-round subkeys. Although this approach works well on SPECK32/64, whose length of subkeys is 16 bits, it may not work efficiently for many other ciphers. For example, in AES-128 [10] the size of subkeys (round keys) and the main secret key is equal to 128 bits, so the complexity of trying all last-round subkeys is equivalent to a brute force attack.

Furthermore, the block size of ciphers can affect the training phase of ML-distinguishers because each bit in the training stage acts as a feature for the machine. SPECK32/64 has a 32 bits block size, but usually, ciphers have a block size higher than that. As a result, training ML-distinguishers could be harder for other block ciphers, especially ones with a SPN structure.

1.2 Contribution

In this paper, we present novel results that significantly advance the efficiency, and reduce the cost of ML-based distinguishers. In particular, we show experimentally that not all the bits in a block are necessary as features to have an adequate neural distinguisher. We also find that different selections of bits (features) in the ML-distinguisher lead to vastly different accuracy results, and that certain bits are consistently better than others for this purpose. On this basis, we propose a new feature selection method for partial differential ML-based distinguishers. We use the selection method to obtain a much more compact, hence easier to learn, partial differential ML-based distinguisher.

In particular, we first present a novel method aimed at training a neural distinguisher more efficiently than in current literature. We do so by introducing the first partial differential machine learning based distinguisher (PD-ML-distinguisher). The idea behind a partial differential ML distinguisher is that it is not necessary to train the ML model on all the bit differences of ciphertext pairs in order to achieve a distinguisher. Consequently, if we have an ML-distinguisher that can tell, without knowing all bits, whether some difference in ciphertext pairs $\delta = C_0 \oplus C_1$ is generated as random or as the result of the encryption of the plaintext pairs, then we do not need to guess all the subkeys in the last round to recover the secret key.

Secondly, we implement the PD-ML-distinguisher for SPECK32/64 and we measure the effectiveness of each bit of δ in the training of ML-distinguishers. Through an extensive series of experiments we find that different bits have a different impact in the training of the distinguisher. This characteristic can be reliably observed in separate, independent experiments. Based on our measurements, we assign a score to each bit with the help of the PD-ML-distinguishers. The bits with higher scores are more important for the training of the models, as they lead to machines that are significantly more effective (68.8% for 8 bits) than those trained on the lowest scoring bits (52% for 8 bits). On this basis, We can therefore select only the most effective bits when training PD-ML-distinguishers, achieving significant efficiencies in the time and space complexity of the distinguisher. This is evidenced by the training of a 6-round distinguisher for SPECK32/64 with just 8 bits achieving an accuracy of 68.8%, against an accuracy of 72% for an equivalent ML distinguisher trained on the full 32 bits.

These results open the way to efficient ML-based differential cryptanalysis of ciphers with larger block sizes, placing standard block ciphers potentially within reach. Additionally, our experimental results on the different impact of different bits can highlight potential weaknesses in block ciphers, and introduces a novel research direction and a new tool for cryptanalists, with implications that go beyond the scope of this work.

1.3 Outline

The structure of this paper is as follows. Section 2 gives an overview of the SPECK32/64 block cipher, and a brief description of Gohr's neural distinguisher is explained. In section 3, PD-ML-distinguishers are introduced, and they are examined on the SPECK32/64 cipher. In section 4, an experiment is presented to measure effectiveness of each bit for training ML-distinguishers, and extensive experimental evidence is discussed.

2 Preliminaries

2.1 The SPECK cipher

A block cipher is a cryptographic algorithm that accepts a fixed-length plaintext, or block, as an input. Then, it applies a transformation function on it to produce a block of encrypted message (ciphertext). The secret key contributes to specifying the transformation function, and as such the security of the block cipher depends on it. A block cipher can be defined by encryption and decryption functions:

$$\begin{aligned} E(P, K) = C : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \\ D(C, K) = P : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n \end{aligned} \quad (1)$$

Where P, C, K, E, D are plaintext, ciphertext, secret key, encryption function and decryption function, respectively.

SPECK is a family of lightweight block ciphers designed by NSA in 2013 [5]. These ciphers have many different block sizes and key sizes, but in this paper, SPECK with 32 bits block size and 64 bits key size is evaluated, and it is specified by SPECK32/64. Like many other block ciphers, it is an iterative cipher, which means it has a function that iterates for many rounds until the ciphertext is generated. The number of rounds for SPECK32/64 is 22.

SPECK is a Feistel cipher. Accordingly, the plaintext is divided into two equal parts (R, L), and in the case of 32 bits block size $R, L \in \{0,1\}^{16}$, then the below function applies to inputs at each round:

$$SPECK32/64: \quad \begin{aligned} L_{r+1} &= ((L_r >> 7) \boxplus R_r) \oplus k_r \\ R_{r+1} &= (R_r << 2) \oplus L_{r+1} \end{aligned} \tag{2}$$

In equation (2) $<<$ and $>>$ are cyclic left and right shift, respectively, $\boxplus$ is modular addition, and $\oplus$ is an exclusive OR (XOR).

2.2 ML-based differential distinguishers

To analyze the security of a block cipher with block size of n against differential attack, cryptographers study the statistical behavior of a *difference* through a block cipher. For this, they choose an input difference and encrypt it for a specific number of rounds and, lastly, try to find non-randomness in corresponding output differences. Throughout this paper, the input difference is represented by $\varDelta$, and it is defined as a XOR of two plaintexts. The output difference is specified by $\delta = E(P_0) \oplus E(P_1 = P_0 \oplus \varDelta)$ and $(\varDelta \to \delta)$ is called a *differential*. The occurrence probability of a differential can be shown by $Pr(\varDelta \to \delta)$. In a random permutation with block size of n, the average probability of a differential is:

$$\forall \varDelta, \delta : Pr(\varDelta \to \delta) = 2^{-n}$$

If an attacker can find a differential that $Pr(\varDelta \to \delta) > 2^{-n}$ for a block cipher, a differential distinguisher is achieved.

In Gohr's attack [12], an ML-based distinguisher is trained with the aid of ciphertext pairs. These pairs are generated in two ways: in the first group, they are real ciphertexts (C_0, C_1) of a block cipher for a specific plaintext pairs (P_0, P_1), where $P_0 \oplus P_1 = \varDelta$; in the second group is selected randomly where $C_0, C_1 \in_r \{0,1\}^n \times \{0,1\}^n$. Then, the accuracy of this distinguisher is evaluated, and if this accuracy is more than 50%, a distinguisher is obtained.

It is important to note that the power and efficacy of this ML-based differential attack compares positively with the All-in-One attack [2] for SPECK32/64. The All-in-One attack is a powerful differential analysis that considers a set of all output differences for a fixed given input difference $\varDelta$ instead of one specific differential trail $\varDelta \to \delta$.

$$All - in - one = \{\delta | \delta = E(P_0) \oplus E(P_0 \oplus \varDelta)\}$$

The steps of finding an ML-based differential distinguisher for r round of SPECK32/64 cipher are as follows. Considering the feistel structure of SPECK32/64, every plaintext can be represented like $P = (L, R)$ that $L \in \{0,1\}^{16}$ and $R \in \{0,1\}^{16}$ are left and right part of plaintext, respectively.

1. First, 10^7 plaintext pairs $(P0, P1)$ are randomly generated in a way that $\Delta = (L_0 \oplus L_1, R_0 \oplus R_1) = (0x0040, 0x0000)$. Meanwhile, 10^7 labels $Y \in_r \{0,1\}^1$ are randomly generated and allocated to the pairs.
2. if $Y = 0$ the $P1$ is randomly changed to $P_1 \in_r \{0,1\}^{32}$ then all these pairs is encrypted with r rounds of SPECK32/64, and all the ciphertext pairs $(C0, C1)$ are stored with their corresponding labels in a dataset.
3. An AI machine is trained with the help of these ciphertext pairs. In this training phase, zero label $Y = 0$ means it is a data from a random permutation, while $Y = 1$ demonstrates a ciphertext for a fixed input difference $\Delta = (0x0040, 0x0000)$.
4. In testing stage, steps (1) and (2) are repeated for another 10^6 pairs, and the accuracy of the machine is measured. If accuracy is more than 50%, then the machine is a differential distinguisher.

As discussed in Section 1.1, following Gohr's seminal work several papers have investigated the use of ML in cryptanalysis [20, 6, 3]. Some of these works changed certain steps of the original attack as mentioned above. For instance, in [3] there are two major changes. Firstly, it is shown that using a multilayer perceptron (MLP) for training the ML-distinguisher can achieve better results than CNN network as used in [12]; and secondly, the training dataset containing $C0 \oplus C1$ instead of $(C0, C1)$ is more useful for increasing the accuracy of ML-distinguisher.

The final goal of having a distinguisher is to attack the cipher. By having a r-round ML-based distinguisher, a trivial attack on $r+1$ rounds of SPECK32/64 cipher can be implemented as follows.

1. For a fixed input difference $\Delta = (0x0040, 0000)$, n pairs of $(P0, P1)$ is formed and their corresponding ciphertext $(C0, C1)_{r+1}$ after $r+1$ rounds is obtained by asking from an oracle.
2. For all possible subkeys in round $r + 1$, (k_{r+1}), ciphertexts are partially decrypted for one round and $(C0, C1)_r$ saved in a dataset.
3. given each $(C0, C1)_r$ to r round ML-based distinguisher in the test phase, a score is attained for every ciphertext pair.
4. Average all the scores to have a final score for each subkey k_{r+1}.
5. Rank the subkeys based on their score. The subkey with the highest score has the most probability to be the correct subkey.

3 Partial Differential ML-distinguisher

In this section, we show that it is possible to train an adequate neural distinguisher based on a subset of bits in a block, rather than the entire block as in previous literature. As a result, we introduce the first Partial Differential ML-distinguisher.

To analyze the security of a cipher, the complexity of the attack algorithm should be less than brute force. The brute force attack complexity for a block cipher is $2^{\min(|k|,|n|)}$, where $|k|$ and $|n|$ are key and block size, respectively. As shown in Section 2.2, in order to find the key of a cipher with the help of a ML-distinguisher, all the subkeys in the last round need to be guessed. Therefore, to attack r rounds of SPECK32/64 cipher, the attacker has to guess all bits of subkey k_r, which has 16 bits length. As a result, the complexity of the attack is 2^{16} which is less than brute force attack 2^{32}, and it is a successful cryptanalysis for SPECK32/64.

On the other hand, in many ciphers, especially those with SPN structures like AES, the complexity of guessing the last round subkey is equal to brute force attack. Hence, in this paper, we train partial differential distinguishers (PD-ML-distinguishers) and compare their accuracy to a full state differential distinguisher for 6-round SPECK32/64. It is shown that PD-ML-distinguishers can still distinguish output differences that are generated by SPECK32/64 from a random output. With the help of these new classifiers, there is no need to guess all the subkeys in the last round, and the complexity of cryptanalysis can be reduced.

3.1 Methodology

In order to show the feasibility of training a machine learning based distinguisher by using partial differences of (P_0, P_1) and (C_0, C_1), we set up an experiment where many PD-ML-distinguishers are trained for 6 rounds of SPECK32/64, and their accuracy is recorded. The number of rounds for encrypting plaintext pairs is set to six rounds. This is consistent with current literature, as in [12], which shows that reasonably strong distinguishers against up to six rounds of Speck can be trained by using ten-layer residual networks. On the other hand, for achieving 7 and 8-round distinguishers, more sophisticated algorithms like *Key Averaging* are required. As our objective is to prove the feasibility and efficiency of a partial differential ML-distinguisher attack compared to a simple ML-distinguisher, rather than increasing the number of rounds that can be attacked using a combination of different techniques, we argue that 6 rounds are sufficient for the scope of this work.

Each bit of the output difference acts as a feature for the machine learning based distinguishers. Consequently, to achieve a partial ML-distinguisher for SPECK32/64, the machine is trained by subset bits of the ciphertext difference, δ, rather than all 32 bits. To experimentally verify to what extent We can trim the output difference, δ, without significantly reducing the effectiveness and robustness of the PD-ML-distinguisher, we conduct an experiment as follows. We first train our distinguisher with just one feature, the least significant bit (LSB) of δ, and record its accuracy. Next, we again train another partial distinguisher, but this time we increase the number of features by one, where the feature is the

Algorithm 1: Training PD-ML differential distinguishers

```
   Input: Data set for training the machines: $\delta = [\delta_0, \ldots, \delta_{31}]$
   Output: Accuracy of machines: $A$
1  $i = 31$, $j = 0$ ;
2  Initialize an empty array $X_{temp}$ ;
3  Initialize array $A$ with size 32;
4  while $i \geq 0$ do
5      $X_{temp} = \delta[i] || X_{temp}$;
6      $D \longleftarrow TrainMachine(\mathrm{X}_{temp})$ ;
7      $A[j] = \mathrm{AccuracyTest}(D)$;
8      $j = j + 1$ , $i = i - 1$
9  end
```

second least significant bit of δ. This process is repeated until we have 32 different distinguishers. Algorithm 1 gives details on the experiment. The plaintext difference Δ to generate ciphertexts difference is $\Delta = (0x0040, 0x0000)$, and the dataset is 32 bits differences $\delta = [\delta_0, \ldots, \delta_{31}]$.

In order to demonstrate the repeatability of the results, we repeat the above process 3 times, each time using a different pairs of (P_0, C_0) and (P_1, C_1) to create new δs, while maintaining $P_0 \oplus P_1 = \Delta = (0x0040, 0x0000)$.

Baski et al. showed in [3] that the Multilayer Perceptron (MLP) architectures are more efficient than Convolutional neural networks (CNN), including Residual networks, or Long Short-Term Memory (LSTM) for training an ML-based differential distinguisher. As a result, in this paper, a Multilayer Perceptron (MLP) machine with three dense layers and a sigmoid activation function is used for training. The number of neurons for dense layers is 32, 64 and 32, respectively. For each number of input bits, a new machine was trained for ten epochs on 10^7 different $C_0 \oplus C_1 = \delta$. Also, another 10^6 sample was generated for validation. The loss function for optimization was binary cross-entropy plus L2 weights regularization with parameter $c = 10^{-5}$ using Adam algorithm [15]. The learning schedule applied in these ML-distinguishers is the cyclic learning rate used in [12]. All other parameters are the default parameters in Keras [9].

In each iteration of Algorithm 1, we concatenate difference bits based on their position. If we assume that the accuracy of the machines entirely depends on the number of bits, rather than which bits are selected, then a different selection method should produce comparable results, accounting for statistical differences. In order to verify this hypothesis, we repeat the above process using Algorithm 1; however, this time, instead of starting from the least significant bit δ_{31}, we start from the most significant bit (MSB) δ_0, and then we concatenate the next MSB to our feature space for the next iteration of the while loop and trained our machine.

The results of the experiment for SPECK32/64 are shown in Figures 1 (LSB$\rightarrow$MSB) and 2 (MSB$\rightarrow$LSB), and are discussed in the following. The results clearly indicate that a partial differential ML-distinguisher is effective, as a dis-

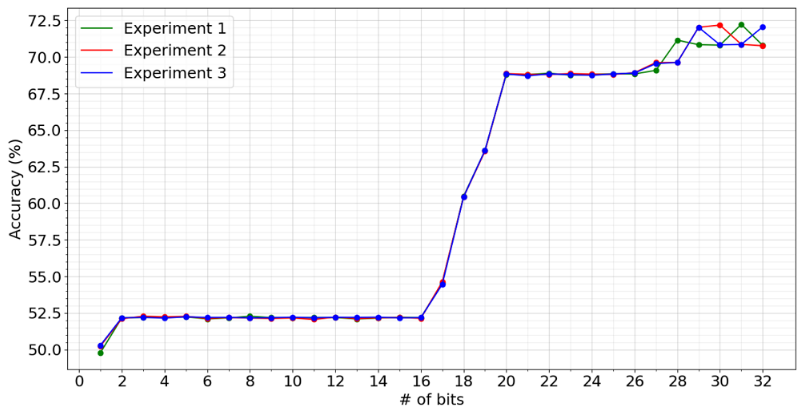

Fig. 1. Accuracy of ML-distinguishers according to number of bits in LSB→MSB direction for 6 round of SPECK32/64

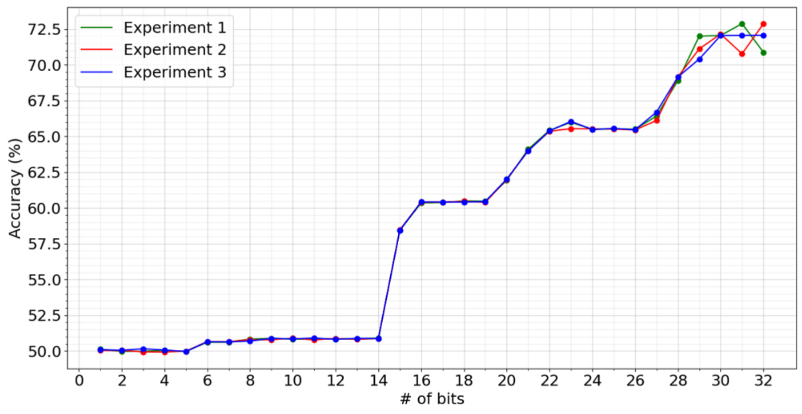

Fig. 2. Accuracy of ML-distinguishers according to number of bits in MSB→LSB direction for 6 round of SPECK32/64

tinguisher can be obtained for a reduced number of bits. However, by looking at the figures, it is also immediately evident that the accuracy changes if we change the selection of bits. These initial experimental results indicate that different bits have a different impact on the machine accuracy, as discussed in the following. We build on these finding in Section 4, where we analyse in detail the impact each bit has in the effectiveness of the partial differential ML-distinguisher.

3.2 Results and Discussion

By looking at Figures 1 and 2, it can be seen that We can achieve a distinguisher without giving all bits of output difference to the machine. For instance, in

Figure 1, We can have a suitable machine with just the first 20 bits of output difference. However, in Figure 2, we train the machines by concatenating the output differences from the opposite MSB to LSB direction. In that case, the number of bits that we need to achieve almost the same accuracy is 28, as shown in Figure 2. From this, We can conclude that the bits position chosen for the PD-ML-distinguisher training can be effective in its accuracy.

We repeat this experiment three times for each direction (LSB to MSB and MSB to LSB) to see how the machines' behavior changes for different datasets. Each color in Figures 1 and 2 represents one run of Algorithm 1. By looking at the figures, it is clear that the results achieved in the experiment can be reliably repeated in different experimental instances. We also note that increasing the number of features (bits) makes the accuracy of PD-ML-distinguishers fluctuate slightly between experiments (for a number of bits > 28). The reason for this is that the number of epochs is set to 10 due to reducing training time, so it is harder for machines to converge when the number of features is higher. Nevertheless, this is not true when the number of bits is lesser than 22 for both directions. Therefore, We can conclude that using PD-ML-distinguishers can cause the machines to converge faster.

4 Measuring bit effectiveness for ML-distinguishers

The results presented in Section 3 show that we do not need all bits of the δ to obtain a ML-based distinguisher. In fact, that We can obtain a partial differential ML distinguisher using a significantly reduced number of bits in the training stage with results comparable to the distinguisher obtained on all bits. In this section, we aim to identify the best strategy for finding the best machine, trained with the least number of bits. Since the desire of block cipher designers is that the output bits of the encrypted message have the most negligible correlation to each other, there is no trivial or pre-defined way to determine which bits are the best for training the partial distinguisher. Hence, we introduce a new experimental method to find the bits of output difference δ for 6-round of SPECK32/64, which have the most impact on the effectiveness of a partial differential ML-distinguisher. The main goal of this experiment is to assign a score to each bit of δ, so that with the help of these scores We can find the most effective bits for training a PD-ML-distinguisher.

4.1 Methodology

The experiment setup is as follows. Given the set of bit positions in a block $B = \{b_0, \dots, b_{31}\}$, where 0 indicates the position of the most significant bit, we select a subset $\mathcal{C}_B^{16} = \{\mathcal{C}_0, \dots, \mathcal{C}_{99}\}$ of the all the possible 16-combinations of B (that is, the subsets of 16 distinct elements of B), in a way that the distribution of each $b_i \in B$, in $\mathcal{C}_i s$ be uniform, where $0 <= i <= 99$. Each of these 100 16-combinations represents a selection of 16 bits out of the 32 total bits in a SPECK32/64 block to be used in the training of a PD-ML-distinguisher.

We use the following procedure in the experiment (Algorithm 2). For each combination, $\mathcal{C}_i \in \mathcal{C}_B^{16}$ we train a PD-ML-distinguisher, and we record its accuracy in an array $A = [a_0, \ldots, a_{99}]$. In the next step, we construct a matrix $M_{32\times100}$ in which every row and column correspond to $b_i s$ and $\mathcal{C}_j s$, respectively. Equation 3 represents the M matrix, and how to construct it.

$$M_{32\times100} = \begin{bmatrix} m_{0,0} & \cdots & m_{0,99} \\ \vdots & \ddots & \vdots \\ m_{31,0} & \cdots & m_{31,99} \end{bmatrix}$$
$$m_{ij} = \begin{cases} 0 & b_j \notin \mathcal{C}_i \\ 1 & b_j \in \mathcal{C}_i \end{cases} \tag{3}$$

This matrix tells us which bits are chosen for every $\mathcal{C}_i s$. For example, if MSB bit is in combination set $\mathcal{C}_0$ then $m_{00} = 0$ otherwise, $m_{00} = 1$. Then, considering the matrix M, We can compute a score for each bit, $S = \{s_0, \ldots, s_{31}\}$, as follows:

$$s_i = \frac{\sum_{j=0}^{99} m_{ij} * a_j}{\sum_{j=0}^{99} m_{ij}} . \tag{4}$$

This score s_i is calculated as the average accuracy of all the PD-ML-distinguishers built on combinations where the ith bit is included. A bit with a higher score means that the combinations with that bit in their set lead to a PD-ML-distinguisher with a higher accuracy on average. As a result, with the help of this score, We can select the most effective bits for training a PD-ML-distinguisher.

4.2 Results and Discussion

By having all the scores, We can select the most effective bits for training a PD-ML-distinguisher with the minimum number of features. Figure 3 shows the scores of each bit for one combination set, $\mathcal{C}_B^{16}$. If the scores shown in Figure 3 were almost similar for all the bits, then we could conclude that all the bits of δs have equal effect on training a PD-ML-distinguisher. However, it is immediately evident there are bits, like 12th, 14th and 29th, with higher scores. As a result, We can confirm this hypothesis that the position of δ bits used for training PD-ML distinguisher affects its accuracy.

In order to demonstrate the repeatability of these results, and since $\mathcal{C}_B^{16}$ is chosen randomly, we repeat the Algorithm 2 three more times and obtain 4 different scores for each bit. Then, we normalize all the scores based on the average for each experiment. Figure 4 shows the result. Also, Table 1 indicates the average score and standard deviation for each bit of δ, considering all scores in the four experiments.

Algorithm 2: Scores of effectiveness

```
Input: Sequence set of combinations: $\mathcal{C}_B^{16} = [\mathcal{C}_0, \dots, \mathcal{C}_{99}]$
Output: Sequence set of scores: $S = [s_0, \dots, s_{31}]$
Training Data: differences of ciphertext pairs of 6-round SPECK32/64 :
        $\delta = [\delta_0, \dots, \delta_{31}]$
1  Initialize Sequence set $A$ with size 100
2  Initialize $M_{32\times 100}$ Matrix
3  Initialize Sequence set $S$ with size 32
4  for $\mathcal{C}_i \in \mathcal{C}_B^{16}$ do
       /* Training PD-ML distinguishers */
5      $D \longleftarrow TrainMachine(\mathrm{C}_i)$
6      $A[i]$ = AccuracyTest($D$)
       /* Making M matrix */
7      for $\delta_j$ in $\delta$ do
8          if $\delta_j \in \mathcal{C}_i$ then
9              $M[j][i] = 1$
10         else
11             $M[j][i] = 0$
   /* Computing the score */
12 for $0 <= i <= 31$ do
13     $S[i] = \frac{\sum_{j=0}^{99} m_{ij} * a_j}{\sum_{j=0}^{99} m_{ij}}$
```

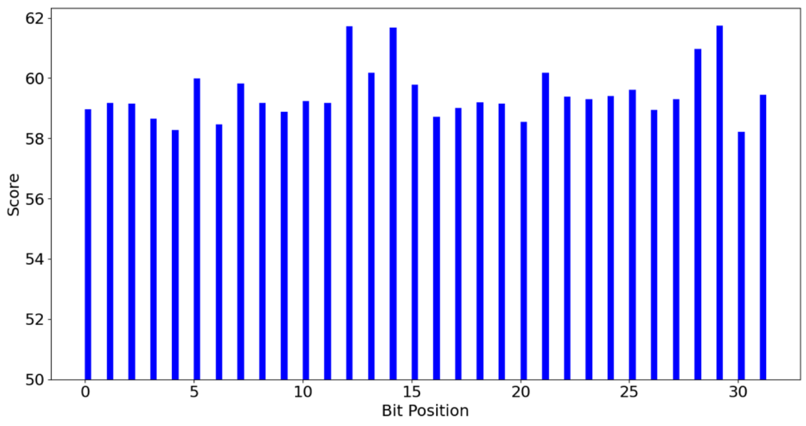

Fig. 3. Bit Scores for one set of combinations $\mathcal{C}_B^{16}$ (experiment 1).

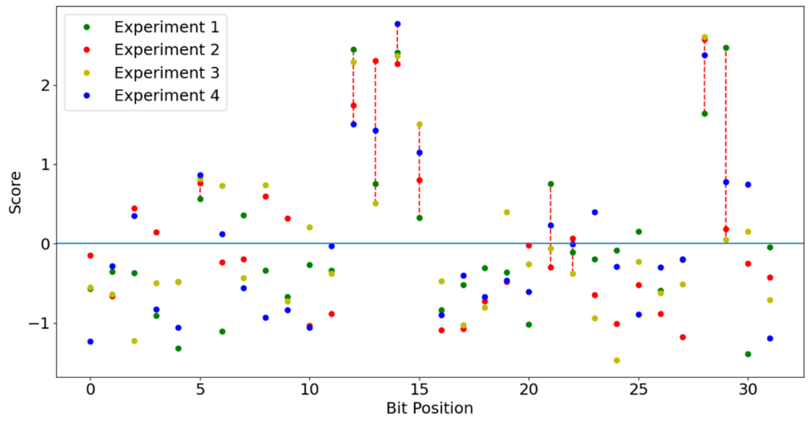

Fig. 4. Normalized Scores for 4 different combination sets. The bits with vertical lines are the ones that are chosen as the most effective bits for training a PD-ML-distinguisher for six rounds of SPECK32/64.

As can be seen, the scores for some bits are always above average. By looking at Figure 4 and Table 1, we choose 8 bits with the best score. We propose two selection of bits for 6-round PD-ML distinguisher of SPECK32/64: $\delta^T = [\delta_{29}, \delta_{28}, \delta_{22}, \delta_{15}, \delta_{14}, \delta_{13}, \delta_{12}, \delta_5]$ and $\delta^{T'} = [\delta_{29}, \delta_{28}, \delta_{21}, \delta_{15}, \delta_{14}, \delta_{13}, \delta_{12}, \delta_5]$. The rationale for choosing these bits is that except bit $\delta_{22} \in \delta^T$ and $\delta_{21} \in \delta^{T'}$ other bits always have a score above average, zero. In the case of δ_{22}, it has a very low standard deviation while its average score is near the total average (zero), while δ_{21} has the next-highest average score across experiments, despite scoring lower than δ_{22} in some instances.

Table 1. The average score (s_i) and standard deviation (SD_i) of each bit of δ in the experiments

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
s_i	-0.62	-0.48	-0.19	-0.51	-0.83	0.75	-0.12	-0.20	0.01	-0.47	-0.53	-0.40	1.99	1.25	2.45	0.94
SD_i	0.38	0.16	0.66	0.41	0.36	0.11	0.66	0.35	0.68	0.46	0.53	0.30	0.38	0.69	0.18	0.43

i	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
s_i	-0.82	-0.75	-0.62	-0.22	-0.47	0.15	-0.10	-0.34	-0.71	-0.36	-0.59	-0.52	2.29	0.87	-0.18	-0.59
SD_i	0.22	0.29	0.19	0.36	0.37	0.39	0.16	0.50	0.55	0.38	0.20	0.39	0.39	0.96	0.78	0.41

Finally, we train a new distinguisher with just these eight bits to verify if they are sufficient for training a neural distinguisher, and we obtain a PD-ML-distinguisher with an accuracy of 68.7% for δ^T and 68.8% for $\delta^{T'}$. For comparison, we train a comparable non-partial differential ML-distinguisher, using all 32 bits, and we obtain an accuracy of 72%. For further comparison and verification, we also train a PD-ML-distinguisher on the lowest scoring 8 bits. This time, we obtain an accuracy of 52%, only slightly above the 50% threshold, at

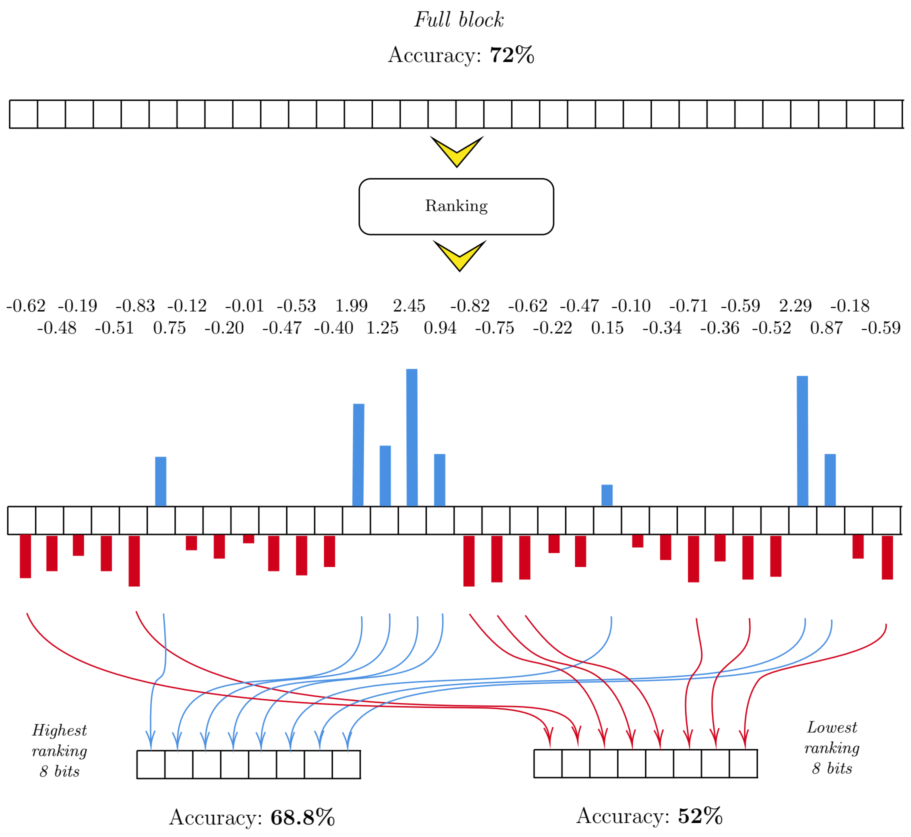

Fig. 5. The Training Process of PD-ML-distinguishers. Rank values are as per Table 1. Plotted bars are approximation of the values (due to graphical constraints).

which the ML model is not able to distinguish from random, and therefore a distinguisher is not obtained. Figure 5 illustrates the process of the training PD-ML-distinguishers.

The above results clearly indicate the validity and effectiveness of the novel partial differential ML-based distinguisher approach we propose for the first time in this paper. The proposed bit-selection mechanism further improves the results, and makes it possible to train a PD-ML-distinguisher using a fraction (25%) of the bits, and therefore leading to a significant reduction in the time and space complexity of training of the model, as well as a reduced size for the neural network. This reduced size means we will have a lesser number of neurons in the input layer. Therefore, it decreases the time complexity of the training phase because We can train the machine with a lower number of epochs compared to when we increase the feature space of the dataset. Moreover, we do not need to guess all the possible subkeys in the key recovery attack; instead, we need to guess just 8 bits of them. We achieve this, while maintaining an effectiveness that is comparable, and only slightly below that of a standard ML-distinguisher built on the full block size.

Although in Sections 3 and 4 we are using the same input difference as Gohr's paper, this bit selection can be used in a black-box method. In [12], a procedure was introduced to find the best input difference Δ for the All-in-One differential attack without any prior knowledge by ML-based distinguishers. So, for having a black-box bit selection, We can use the ML-distinguisher for finding the best possible Δ and then use the bit selection of this section.

The details of the ML models used are summarised in Table 2, showing that our Partial ML-distinguisher machine is significantly smaller in terms of depth, number of epochs and number of features than the comparable Gohr's machine [12]. A marginal reduction in the accuracy (69% for 8-bit, against 78%) is compensated by the fact that the reduced number of features in the machine directly translate in a reduction in effort for key recovery attacks, in particular if the attack is based on the key ranking. This is because we only need to guess 8 bits of the subkey of the last round rather than the full state of the subkey.

Table 2. Comparison of different machine learning based differential distinguishers

Reference	Cipher	Rounds	Network	# of input features	Depth	Epochs
[12]	Speck32/64	6	CNN	32 (bits)	10	200
This paper	Speck32/64	6	MLP	32 (bits)	3	10
This paper	Speck32/64	6	MLP	8 (bits)	3	10

5 Conclusion

In this paper, we investigated the applicability of partial differences in training neural distinguishers, and we proposed the first partial differential Machine Learning (ML) distinguisher. As a partial differential ML-distinguisher is trained on a selection of bits rather than all bits in a block, we also studied the impact of the selection of bits in the accuracy of the distinguisher, and we established that certain bits have reliably higher effectiveness than others, through a series of independent experiments on different inputs. On this basis, we proposed an algorithm for assigning an effectiveness score to each bit in the block.

In applying a differential attack, our goal was to find non-random behavior in a block cipher when we do not have all bits of difference in ciphertext pairs, δ. To achieve this purpose, we trained a ML-based differential distinguisher for 6-rounds of SPECK32/64 by using just some parts of δ, and we studied the effectiveness of such a partial ML-based differential distinguisher, which we call PD-ML-distinguisher. Our experiments indicate that it is possible to achieve PD-ML-distinguishers with high accuracy, that is comparable to that of a ML-distinguisher trained on the full 32-bits of the block. We also observe that increasing the number of bits does not necessarily lead to an increase in the accuracy of the machine; also, raising the number of bits can reduce the converging speed.

Based on these results, we then experimentally examined if all bits have an equal impact on the training stage by creating new PD-ML-distinguishers. We detect that the accuracy changes when we alter the bits. We conclude that some bits are more critical for ML-distinguishers.

To find the most effective bits in the training phase, we proposed an algorithm to allocate a score to each bit of δ for 6-round SPECK32/64. With the help of this score, we could select the eight most effective bits and construct a 6-round PD-ML-distinguisher for SPECK32/64 achieving an accuracy of 68.8%. This is comparable, and only slightly lower than the 72% accuracy of a ML-distinguisher trained on the full 32-bits, at a significantly higher cost in terms of computational and space complexity. The reduced input also leads to a significant reduction in the complexity of a potential subsequent key recovery attack, as we would not need to guess all the possible subkeys, but rather just 8 bits of them.

While our experiments are obtained using lightweight cipher SPECK32/64, the proposed techniques are generic and can be applied on other ciphers. These results are in fact likely to open the way to efficient ML-based differential cryptanalysis of ciphers with larger block sizes, placing standard block ciphers potentially within reach.

The bit effectiveness scores that are presented in this paper for the first time, provide a novel metric and additional information on the inner functioning of a cipher. The scores are potentially indicating a new path to discovery of non-random behavior and associated weaknesses in block ciphers, with applications well beyond the scope of this work.

References

1. Abadi, M., Andersen, D.G.: Learning to protect communications with adversarial neural cryptography. arXiv preprint arXiv:1610.06918 (2016)
2. Albrecht, M.R., Leander, G.: An all-in-one approach to differential cryptanalysis for small block ciphers. In: Knudsen, L.R., Wu, H. (eds.) Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7707, pp. 1–15. Springer (2012). https://doi.org/10.1007/978-3-642-35999-6_1
3. Baksi, A., Breier, J., Chen, Y., Dong, X.: Machine learning assisted differential distinguishers for lightweight ciphers. In: 2021 Design, Automation & Test in Europe Conference & Exhibition (DATE). pp. 176–181. IEEE (2021)
4. Bassham, L., Çalık, Ç., McKay, K., Turan, M.S.: Submission requirements and evaluation criteria for the lightweight cryptography standardization process. US National Institute of Standards and Technology (2018)
5. Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The simon and speck lightweight block ciphers. In: Proceedings of the 52nd Annual Design Automation Conference. pp. 1–6 (2015)
6. Bellini, E., Rossi, M.: Performance comparison between deep learning-based and conventional cryptographic distinguishers. IACR Cryptol. ePrint Arch. **2020**, 953 (2020), https://eprint.iacr.org/2020/953

7. Benamira, A., Gerault, D., Peyrin, T., Tan, Q.Q.: A deeper look at machine learning-based cryptanalysis. IACR Cryptol. ePrint Arch **287**, 2021 (2021)
8. Biham, E., Shamir, A.: Differential cryptanalysis of the full 16-round des. In: Annual International Cryptology Conference. pp. 487–496. Springer (1992)
9. Chollet, F., et al.: Keras. https://github.com/fchollet/keras (2015)
10. Daemen, J., Rijmen, V.: Aes proposal: Rijndael (1999)
11. Fu, K., Wang, M., Guo, Y., Sun, S., Hu, L.: Milp-based automatic search algorithms for differential and linear trails for speck. In: International Conference on Fast Software Encryption. pp. 268–288. Springer (2016)
12. Gohr, A.: Improving attacks on round-reduced speck32/64 using deep learning. In: Annual International Cryptology Conference. pp. 150–179. Springer (2019)
13. Gurobi Optimization, L.: Gurobi optimizer reference manual (2021), http://www.gurobi.com
14. Hou, B., Li, Y., Zhao, H., Wu, B.: Linear attack on round-reduced DES using deep learning. In: European Symposium on Research in Computer Security. pp. 131–145. Springer (2020)
15. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
16. Liu, G., Lu, J., Li, H., Tang, P., Qiu, W.: Preimage attacks against lightweight scheme xoodyak based on deep learning. In: Arai, K. (ed.) Advances in Information and Communication, FICC 2021. vol. 1364, pp. 637–648. Springer International Publishing, Cham (2021)
17. Matsui, M.: Linear cryptanalysis method for des cipher. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 386–397. Springer (1993)
18. Mironov, I., Zhang, L.: Applications of sat solvers to cryptanalysis of hash functions. In: International Conference on Theory and Applications of Satisfiability Testing. pp. 102–115. Springer (2006)
19. Schneier, B.: Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley; 20th edition (2015)
20. Yadav, T., Kumar, M.: Differential-ml distinguisher: Machine learning based generic extension for differential cryptanalysis. In: International Conference on Cryptology and Information Security in Latin America. pp. 191–212. Springer (2021)
21. Zahednejad, B., Li, J.: An improved integral distinguisher scheme based on deep learning. Tech. rep., EasyChair (2020)