electronics

MDPI

Article

Deep-Learning-Based Neural Distinguisher for Format-Preserving Encryption Schemes FF1 and FF3

Dukyoung Kim, Hyunji Kim, Kyungbae Jang, Seyoung Yoon and Hwajeong Seo *

Division of IT Convergence Engineering, Hansung University, Seoul 02876, Republic of Korea; dithr3@hansung.ac.kr (D.K.); 1594012@hansung.ac.kr (H.K.); starj1234@hansung.ac.kr (K.J.); yoonsy@hansung.ac.kr (S.Y.)

* Correspondence: hwajeong@hansung.ac.kr; Tel.: +82-760-8033

Abstract: Distinguishing data that satisfy the differential characteristic from random data is called a distinguisher attack. At CRYPTO'19, Gohr presented the first deep-learning-based distinguisher for round-reduced `SPECK`. Building upon Gohr's work, various works have been conducted. Among many other works, we propose the first neural distinguisher using single and multiple differences for format-preserving encryption (FPE) schemes `FF1` and `FF3`. We harnessed the differential characteristics used in `FF1` and `FF3` classical distinguishers. They used SKINNY as the inner encryption algorithm for `FF3`. On the other hand, we employ the standard `FF1` and `FF3` implementations with AES encryption (which may be more robust). This work utilizes the differentials employed in `FF1` and `FF3` classical distinguishers. In short, when using a single `0x0F` (resp. `0x08`) differential, we achieve the highest accuracy of 0.85 (resp. 0.98) for `FF1` (resp. `FF3`) in the 10-round (resp. 8-round) number domain. In the lowercase domain, due to an increased number of plaintext and ciphertext combinations, we can distinguish with the highest accuracy of 0.52 (resp. 0.55) for `FF1` (resp. `FF3`) in a maximum of 2 rounds. Furthermore, we present an advanced neural distinguisher designed with multiple differentials for `FF1` and `FF3`. With this sophisticated model, we still demonstrate valid accuracy in guessing the input difference used for encryption.

Keywords: deep learning; distinguisher; differential characteristic; format-preserving encryption

Citation: Kim, D.; Kim, H.; Jang, K.; Yoon, S.; Seo, H. Deep-Learning-Based Neural Distinguisher for Format-Preserving Encryption Schemes FF1 and FF3. *Electronics* **2024**, *13*, 1196. https://doi.org/10.3390/electronics13071196

Academic Editors: Yongjun Ren and Hu Xiong

Received: 26 February 2024
Revised: 21 March 2024
Accepted: 22 March 2024
Published: 25 March 2024

1. Introduction

Differential cryptanalysis [1] is one of the primary cryptanalysis techniques. If it is possible to predict the key by analyzing the differential characteristic, the cryptographic algorithm can be considered insecurely designed. Distinguishing data that satisfy differential characteristics (i.e., input/output differentials) from random data is referred to as a distinguisher attack, which is more powerful than an exhaustive search. Depending on the method of classifying the input difference, it is divided into a binary classification model (i.e., distinguishing random data and input difference) and a multi-classification model (i.e., distinguishing multiple-input differences).

Recently, with the development of deep learning [2–8], various studies on deep-learning-based distinguishers [9,10] have been presented [11–21]. Deep learning is well-suited for probabilistically distinguishing data that satisfy differential characteristics, as it has the capability to make probabilistic predictions about data. In detail, the deep learning algorithm [22] consists of multiple layers, each composed of multiple neurons. Neurons calculate their final values by summing the weighted values from the previous layer and passing them through an activation function. This process is repeated for each layer, starting from the input layer. The network learns by minimizing the difference between the predicted output and the actual labels using a loss function (e.g., binary cross-entropy, categorical cross-entropy, mean squared error, etc.). In this process, an optimization function (e.g., stochastic gradient descent (SGD), RMSprop, Adam) is used for effective minimization.

Once trained, the network can predict using its trained weights. A well-trained network can make robust predictions even about untrained data, and the design goal is to create such a robust neural network. For this reason, many studies on neural distinguishers are being conducted, but research on deep-learning-based distinguishers for format-preserving encryption (FPE) schemes ([23], see Section 2.1 for details) such as `FF1` (https://github.com/PaddyKe/FFX/tree/main/FFX/FF1 accessed on 21 March 2024) and `FF3` (https://pypi.org/project/ff3/ accessed on 21 March 2024), has not yet been conducted.

1.1. Our Contribution

In this work, for the first time, we propose a neural distinguisher based on deep learning for FPE schemes `FF1` and `FF3` considering single- and multiple-input differences. Significantly, our results demonstrate that the deep-learning-based distinguisher is well suited for format-preserving encryption schemes as well. In brief, the following contributions are presented in this paper:

1. The first neural distinguisher for the NIST FPE family: We propose the first neural distinguisher for `FF1` and `FF3`, which are NIST standard format-preserving ciphers. Our neural distinguisher works successfully in the number and lowercase domains and can be effectively utilized for cryptanalysis using differential characteristics.
2. Successful verification of two models (single-input difference and multiple-input difference): Our neural distinguisher is divided into an implementation that distinguishes single-input differences and multiple-input differences. When using a single-input difference, the cipher data are distinguished from random data. When multiple-input differences are used, the model can distinguish the input difference used for the input data among multiple-input differences. We adopt both approaches and successfully demonstrate the effectiveness of our model.
3. Our neural distinguisher can attack various variants of `FF1` and `FF3`: While format-preserving encryption includes an encryption function, the presence of differential characteristics remains independent of the specific encryption function. Consequently, our neural distinguisher can be effectively employed for distinguisher attacks targeting various variants of `FF1`, `FF3`.

1.2. Organization

The remainder of this paper is organized as follows. The backgrounds of the format-preserving encryption and the neural distinguisher are summarized in Section 2. In Section 3, we present the first neural distinguisher (single- and multiple-input differences) for `FF1` and `FF3`. We evaluate and analyze the performance of our neural distinguisher for `FF1` and `FF3` in Section 4. Finally, we discuss and conclude this work in Section 5.

2. Prerequisites

2.1. Format-Preserving Encryption

When applying block ciphers to database encryption, it often leads to changes in the data type or length, necessitating database structure engineering. This issue becomes particularly critical when encrypting sensitive data such as credit card numbers.

However, format-preserving encryption (FPE) [23] is a method that preserves the plaintext structure even after encryption, unlike block ciphers. As a result, there is no need for additional storage capacity to store ciphertext compared to plaintext. In this context, FPE is a cost-effective and efficient solution for integration into database systems without requiring extensive engineering efforts.

In this work, our focus is on the FPE schemes `FF1` and `FF3`, both standardized by NIST (https://csrc.nist.gov/news/2016/nist-released-special-publication-800-38g accessed on 21 March 2024). `FF1` consists of 10 rounds with the same block size and a key size of 128 bits, while `FF3` comprises 8 rounds with a block size of 32 bits and a key size of 128 bits. Both FPE ciphers are designed using a Feistel architecture and incorporate encryption functions

similar to AES into the inner round function (it is worth noting that the encryption function used within FPE can be customized [24,25]).

Although FF1 and FF3 share some similarities, FF1 offers higher security due to its increased number of rounds and its ability to support a wider range of protected data formats compared to FF3. On the contrary, FF3 has a higher data throughput compared to FF1.

2.2. Differential Characteristic

Differential cryptanalysis [1] is a representative cryptanalysis method of block ciphers. The input difference (δ) is the XOR between the plaintext pairs (P_0, P_1), and the output difference (Δ) is the XOR between the ciphertext pairs. As in Equation (1), C_0 and C_1 are the results of encrypting (E) P_0 and P_1 , respectively. The output difference (Δ) can be obtained by XORing C_0 and C_1. Here, a differential characteristic means a pair of input and output differences (δ, Δ).

In the case of an ideal cipher, when plaintext with any input difference is encrypted, the output difference should be uniform (like random). A weak cryptographic algorithm has a certain output difference corresponding to an input difference. If the probability of satisfying an output difference for an input difference is greater than the random probability, the ciphertext can be distinguished from the random. These characteristics have remained even when encryption is performed and can be inferred probabilistically.

$$\begin{aligned} &P_1 = P_0 \oplus \delta, \\ &C_0 = E(P_0), C_1 = E(P_1), \\ &\Delta = C_0 \oplus C_1 \end{aligned} \tag{1}$$

2.3. Neural-Network-Based Distinguisher for Differential Cryptanalysis

A neural network can be a good solution for distinguisher attacks, as it can probabilistically satisfy specific output differences for given input differences. Consequently, the neural distinguisher performs probabilistic prediction on data applied to distinguisher attacks using differential characteristics. Most of the ongoing works of neural distinguishers are derived from [11], and they focus on target ciphers and input differences. In [11], proposed at CRYPTO'19, the first neural distinguisher is proposed for round-reduced SPECK32/64. Their neural distinguisher successfully distinguishes cryptographic data from random data for up to 7 rounds and extends up to 8 rounds through transfer learning. In [12], two distinguisher models considering multi-input differential and single differential are presented. And the target ciphers are GIMLI, ASCON, KNOT, and Chaskey. The proposed MLP-based neural distinguisher successfully distinguishes 8-round GIMLI, 3-round ASCON, 10/12-round KNOT (256/512-bit), and 4-round Chaskey. In addition, many works [13,17–20] on various cryptographic and differential characteristics are being conducted, focusing on SPECK.

3. Neural Distinguisher for FF1 and FF3

This section describes our neural distinguisher specifically designed for the FPE schemes (FF1 and FF3). Our neural distinguisher is based on the Baksi et al. scheme [12]. Also, our neural distinguisher for FPE schemes is based on Dunkelman et al.'s ePrint'20 paper [26]. They determined the differential characteristic of FPE shemes. Furthermore, our implementation is categorized into two types based on the utilized input differences, namely, *ModelOne* (Algorithm 1) and *ModelMul*.

ModelOne is a binary model capable of distinguishing cipher data with a single-input difference from random data, while *ModelMul* is designed to distinguish multiple-input differences. Details about both models are described in Sections 3.1 and 3.2. In addition, we perform the hyper-parameter optimization for both models.

Algorithm 1 *ModelOne*: Training procedure

```
Training Data TD ← [ ]                                   ▷ Empty state
for i from 0 to n − 1 do
    Choose random plaintext P0 and P1
    P2 ← P0 ⊕ δ
    Ciphertexts C0, C1, and C2 ← FPE_enc (P0, P1, and P2)   ▷ Generate ciphertexts
    TD_i ← Assign labels 0 to (C0||C1) and 1 to (C0||C2)
end for
Train model DL with TD
a ← Output of DL                                         ▷ a is training accuracy
if a > 1/2 then
    Continue the training procedure
else                                                     ▷ a = 1/2
    Abort DL
end if
```

3.1. ModelOne: Single-Input Difference

3.1.1. Dataset

Figure 1 illustrates the overall generation process and the generated dataset using a single-input difference of *ModelOne*. First, random plaintexts P_0 and P_1 for encryption are generated. Furthermore, we generate plaintext P_2 to satisfy the input difference δ with P_0 (i.e., $P_2 = P_0 \oplus \delta$). Then, the ciphertexts C_0, C_1, and C_2 are generated by encrypting the plaintexts P_0, P_1, and P_2.

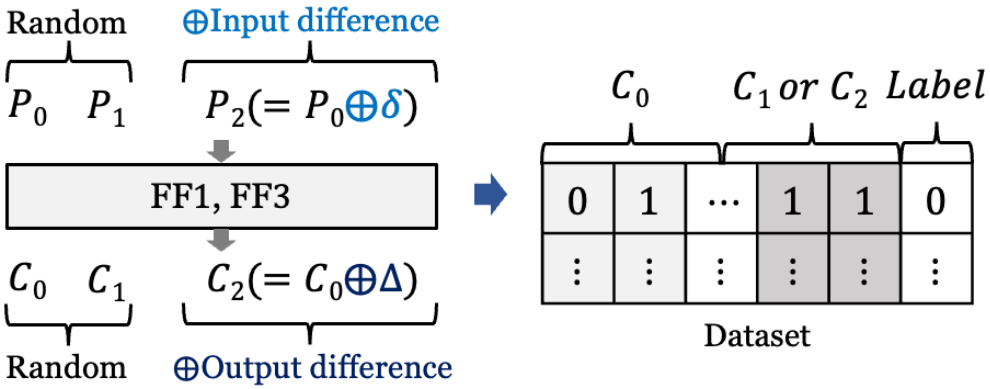

Figure 1. Dataset with one input difference.

C_0 and C_1 are the ciphertexts generated by encrypting the random plaintexts P_0 and P_1 that do not satisfy a differential characteristic (i.e., $\notin \delta - \Delta$). On the other hand, the pair of C_0 and C_2 has a special relationship that satisfies the differential characteristic (i.e., $\in \delta - \Delta$).

We assign the label 0 (random) to the result of concatenating the two values $(C_0||C_1)$ indicating random data. On the other hand, C_0 and C_2 are the ciphertexts for plaintexts that satisfy the input difference δ. The concatenated value $(C_0||C_2)$ corresponds to cipher-related data that satisfy the differential $\delta - \Delta$ with a certain probability. We assign label 1 (cipher) to the concatenated result of C_0 and C_2 $(C_0||C_2)$, which satisfies the differential.

FPE is designed to operate within specific domains, ensuring that encrypted data maintain their original format, which is crucial for data integrity, compliance, and system compatibility. Throughout this paper, we define the following two domains: the number domain (`0` to `9`) and the lowercase-letter domain (`a` to `z`). Also, the dataset consists of bits of ciphertext pairs (i.e., $C_0||C_1 \rightarrow 0...1||0...0$).

For the input difference (for the dataset), we use `0x0`$||K$ (K is a hexadecimal number ranging from `0x0` to `0xF`). Our choice is based on Equation (3) in [26]. The authors demonstrate that when `0x0`$||K$ is used, the probability of a differential is high. It should be

noted that since these input differences are independent of the inner encryption function (such as `SKINNY`, `SPECK`, or `AES`), our work can be applied to various implementations of FPE schemes.

3.1.2. Architecture and Training

ModelOne receives concatenated random data ($C_0||C_1$) or cipher data ($C_0||C_2$) and classifies them into random (label 0) or cipher (label 1). Each bit of the ciphertext pair in the dataset is assigned to each neuron of the input layer. Then, the output of the input layer passes through the hidden layer. In the output layer, a final value between 0 and 1 is calculated by applying a sigmoid activation function. Then, the loss of the final value and the actual value (0 or 1) is calculated. Figure 2 shows the process of *ModelOne* using a single-input difference.

If training to distinguish input data is performed correctly, our model can work as a neural distinguisher for `FF1` and `FF3`. To work as a valid distinguisher, it must achieve an accuracy greater than $\frac{1}{2}$, which is a random probability.

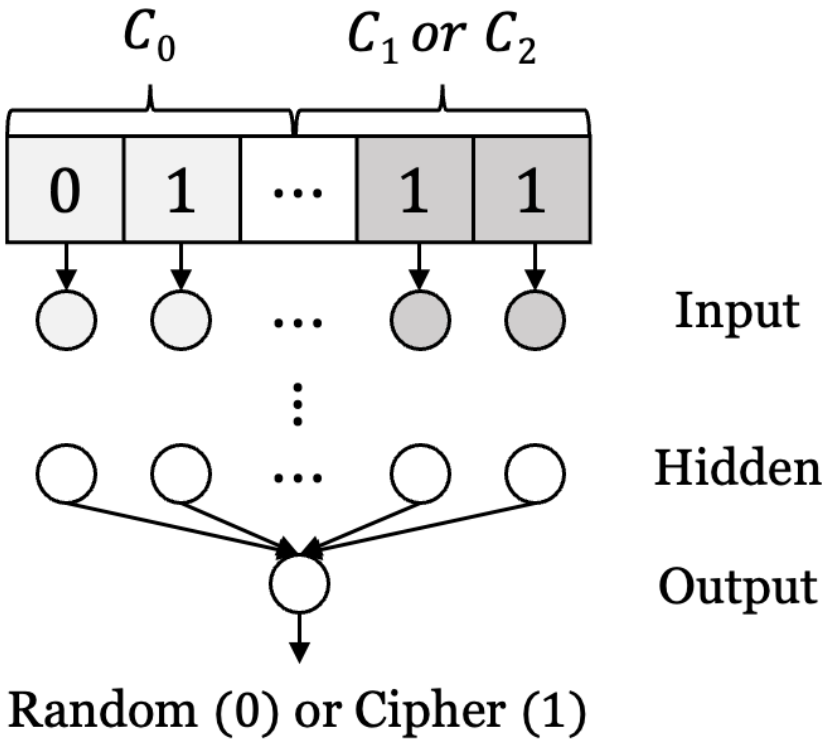

Figure 2. System diagram of *ModelOne*.

Table 1 shows the hyperparameters of *ModelOne* (`FF1` and `FF3`). The epoch is set to 20 and 15 for *ModelOne*, and a dense layer with all nodes fully connected is used. *ModelOne* performs binary classification because it should distinguish input from random or cipher data. Thus, binary cross-entropy is used as the loss function. Additionally, the Adam optimization function (known for its good performance) is employed in our model. For more sophisticated learning, the learning rate of the optimization function is adjusted during training (the learning rate starts at 0.001 and decreases to 0.0001).

Table 1. Hyperparameters of *ModelOne* and *ModelMul*.

Model	*ModelOne*	*ModelMul*
Schemes	FF1/FF3	FF1/FF3
Epochs	20/15	20/15
Loss function	Binary cross-entropy	Categorical cross-entropy
Optimizer	Adam (0.001 to 0.0001, learning rate decay)	
Activation function	ReLu (hidden)	
	Softmax (output)	Sigomid (output)
Batch size	32	
Hidden layers	5/4 hidden layers (with 64/128 units)	
Parameters	173,956/74,497	173,956/75,787

3.2. ModelMul: Multiple-Input Differences

3.2.1. Dataset

Similar to *ModelOne*, a random plaintext P_0 is generated. Then, plaintext pairs that satisfy multiple-input differences are generated. That is, P_0 is XORed with δ_n (different input difference) to obtain plaintext P_n. Lastly, each plaintext P_n (with different input differences) is encrypted to generate the ciphertext C_n. In short, *ModelMul* takes multiple ciphertexts with different input differences as a training data set.

$C_0||C_n$ is labeled as class $n-1$ since C_n is the ciphertext obtained by encrypting the plaintext with n different input differences, respectively (e.g., C_3 corresponds to Δ_3). In the distinguisher that uses multiple-input differences, the number domain (0 to 9) and the lowercase-letter domain (a to z) are also used in the FF1 and FF3 encryption process. As in *ModelOne*, we adopt the input difference 0x0||K (K is a hexadecimal number ranging from 0x0 to 0xF). Figure 3 shows the generation process of the dataset using multiple input differences.

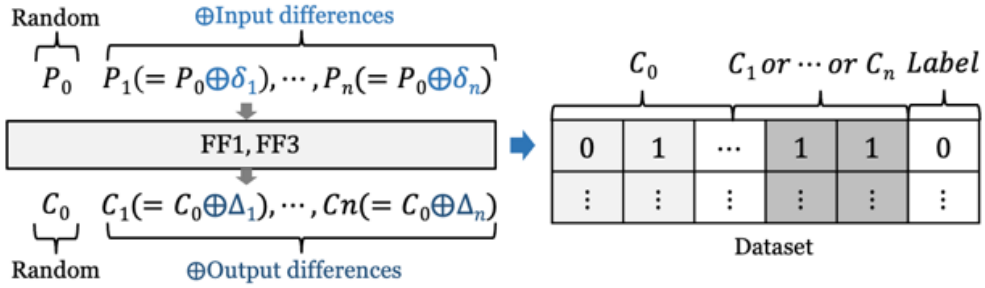

Figure 3. Dataset with multiple-input differences.

3.2.2. Model Architecture and Training

In this model, the attacker chooses the input differences δ_0, δ_1,..., δ_{n-1} ($n > 2$). Figure 4 and Algorithm 2 show the system logic of *ModelMul* using multiple-input differences. In the training phase, the deep learning model is learning to find whether there is any pattern (i.e., differential characteristics) in the outputs. Through this training process, *ModelMul* can distinguish multiple-input differences. While *ModelOne* can classify only random and one-input differences, *ModelMul* works as a distinguisher for data that satisfy multiple differential characteristics. If n input differences are used, an accuracy greater than $1/n$ (the probability of random data) must be achieved in order to work as a valid distinguisher. If the accuracy of the training is higher than $1/n$, the model finds a pattern from the cipher's outputs and a differential attack proceeds. On the other hand, if the training accuracy is less than or equal to $1/n$, the model is aborted.

In brief, *ModelMul* receives ciphertext pairs that satisfy the differential characteristics as input and classifies them based on the input differences used. Finally, our *ModelMul* can distinguish the input differences used in the cipher data.

Algorithm 2 *ModelMul*: Training procedure

Training Data $TD \leftarrow [\ ]$ ▷ Empty state
Choose random plaintext P ▷ Step 2
Ciphertext $C \leftarrow FPE_{enc}(P)$ ▷ FPE_{enc} means FF1 or FF3 encryption
for i from 0 to $n-1$ **do**
    $P_i \leftarrow P \oplus \delta_i$
    $C_i \leftarrow FPE_{enc}(P_i)$
    Append TD with $(C_i \oplus C, i)$ ▷ $C_i \oplus C$ is from class i
end for
Repeat from Step 2
Train DL model with TD
$a \leftarrow$ Output of trained DL model ▷ a is training accuracy
if $a > \frac{1}{n}$ **then**
    Continue the training procedure
else ▷ $a = \frac{1}{n}$
    Abort DL model
end if

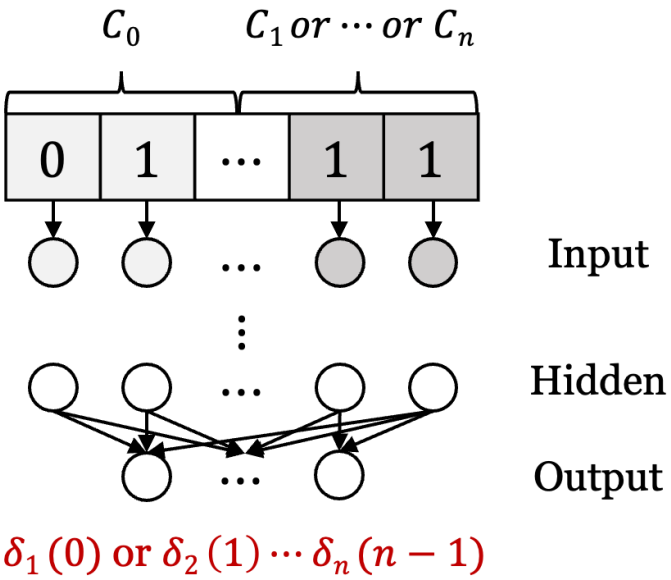

Figure 4. System diagram of *ModelMul*.

3.3. Hyper-Parameter Tuning

Table 1 also lists the hyperparameters (FF1 and FF3) of ModelMul. Initially, we set the epoch to 50, but it was confirmed that the same accuracy was achieved even at 20 and 15, so the optimal epoch is set. Also, for *ModelMul*, the same dense layer as *ModelOne* is used. However, *ModelMul* performs multi-class classification because it classifies multiple pairs of ciphertexts that satisfy the output difference. In addition, the Adam optimization function is also used. Lastly, for more sophisticated learning, the optimizer learning rate is adjusted from 0.001 to 0.00001 and the Relu function is used for fast convergence to the hidden layer. We experimented with various batch sizes (e.g., 64, 128) but confirmed that 32 is optimal in terms of memory, time, and accuracy. In addition, we tried to adjust the number of layers and the number of units in each layer, but it can be seen that the complex model is not efficient for our dataset.

As noted earlier, our model structures are simple, but they are optimized models for our dataset. There is an advantage in terms of file size, as the model structure is not

complicated. In addition, our models have been sufficiently simplified such that there will be no shortage for deployment. If it is further optimized and made lightweight, it will have more advantages in large-scale deployment. Therefore, this remains our future work. In particular, we built a deep learning model that can be applied to both *ModelOne* and *ModelMul*. This shows that it is a model with high generalization performance.

4. Evaluation

4.1. Experimental Environment

This experiment is performed on Google Colaboratory, a cloud computing platform supporting Ubuntu 20.04.5 LTS and Tesla T4 (GPU) 12GB RAM. As the programming environment, TensorFlow 2.12.0 and Python 3.9.16 are used.

To enhance the robustness of our result, we deliberately constructed a separate dataset to serve as an independent variable in our experimentation. Furthermore, by conducting multiple trials, we significantly increased the reliability of our results. This methodological approach ensures that the outcomes observed are not merely coincidental but are, in fact, the average of numerous iterations.

4.2. Result for One-Input Difference

In the case of `FF1` (resp. `FF3`), in the number domain, when utilizing `0x0F` (resp. `0x08`) as the input difference, *ModelOne* can effectively distinguish data for up to 10 rounds (resp. 8 rounds) with a commendable accuracy of 0.85 (resp. 0.98). When employing different input differences, it demonstrates relatively lower accuracy compared to `0F` (resp. `08`)

In the lowercase domain, as the number of plaintext and ciphertext cases increases, *ModelOne* for `FF1` (resp. `FF3`) can distinguish data for a maximum of 2 rounds. It attains an accuracy of 0.522 (resp. 0.55) for `0x0F` (resp. `0x08`), which is somewhat lower than in the number domain. The input difference `0x03` (resp. `0x01`) provides an accuracy approximately 0.1 (resp. 0.35) lower than that of `0x0F` (resp. `0x08`). The underlying reason for these results, as noted in [26], is that when employing input differences of `0x0F` (resp. `0x08`), they exhibit the anticipated differentiation properties. Tables 2 and 3 show the results of `FF1` and `FF3` *ModelOne* based on input differences. This experiment reaffirms that data with output differences based on the input differences of `0x0`$||K$ can be reliably predicted with a high probability.

Table 2. Result of `FF1` *ModelOne* according to input difference.

0x	Number (10 Rounds)				Lowercase (2 Rounds)			
	Training	Validation	Test	Reliability	Training	Validation	Test	Reliability
01	0.732	0.741	0.733	0.233	0.500	0.500	0.500	0.000
02	0.741	0.752	0.743	0.243	0.510	0.512	0.510	0.010
03	0.711	0.712	0.711	0.211	0.522	0.520	0.522	0.022
04	0.751	0.752	0.752	0.252	0.511	0.512	0.510	0.010
05	0.752	0.751	0.752	0.252	0.511	0.512	0.511	0.011
06	0.751	0.752	0.752	0.252	0.511	0.512	0.511	0.011
07	0.751	0.751	0.752	0.252	0.511	0.511	0.511	0.011
08	0.801	0.802	0.802	0.302	0.511	0.511	0.511	0.011
09	0.841	0.842	0.841	0.341	0.522	0.521	0.522	0.022
0A	0.842	0.841	0.841	0.341	0.500	0.510	0.510	0.010
0B	0.822	0.821	0.822	0.322	0.511	0.511	0.511	0.011
0C	0.855	0.854	0.855	0.355	0.500	0.500	0.500	0.000
0D	0.788	0.788	0.788	0.288	0.511	0.511	0.511	0.011
0E	0.811	0.812	0.811	0.311	0.522	0.521	0.522	0.022
0F	0.855	0.854	0.855	0.355	0.522	0.522	0.522	0.022

Table 3. Result of FF3 *ModelOne* according to input difference.

0x	Number (8 Rounds)				Lowercase (2 Rounds)			
	Training	Validation	Test	Reliability	Training	Validation	Test	Reliability
01	0.629	0.624	0.623	0.123	0.545	0.544	0.543	0.043
02	0.829	0.825	0.825	0.325	0.552	0.548	0.545	0.045
03	0.783	0.769	0.771	0.271	0.52	0.514	0.513	0.013
04	0.761	0.756	0.757	0.257	0.523	0.52	0.517	0.017
05	0.773	0.752	0.747	0.247	0.539	0.538	0.537	0.037
06	0.758	0.748	0.75	0.25	0.523	0.519	0.523	0.023
07	0.756	0.739	0.74	0.24	0.532	0.529	0.529	0.029
08	0.987	0.976	0.977	0.477	0.556	0.554	0.554	0.054
09	0.962	0.942	0.941	0.441	0.547	0.543	0.549	0.049
0A	0.969	0.953	0.951	0.451	0.538	0.534	0.532	0.032
0B	0.97	0.965	0.966	0.466	0.53	0.526	0.522	0.022
0C	0.97	0.959	0.959	0.459	0.538	0.536	0.539	0.039
0D	0.968	0.965	0.966	0.466	0.532	0.524	0.518	0.018
0E	0.964	0.963	0.963	0.463	0.549	0.549	0.551	0.051
0F	0.965	0.939	0.941	0.441	0.528	0.524	0.524	0.024

4.3. Result for Multiple-Input Differences

We use the input difference $0x0||K$ (see Section 3.1). Each dataset is set according to the input difference pair used and there are $2^{18.6097}$ data points in each class. We set 0x0F (resp. 0x08), assumed as the best difference in FF1 (resp. FF3) [26], as a fixed input difference. Then, with 0x0F (resp. 0x08) fixed, datasets are generated by expanding the data for different input differences. Table 4 shows the details of the input difference dataset for *ModelMul*.

Table 4. Details of the input difference dataset.

Dataset	Data Size	Input Difference Pair	Valid Accuracy
I1	$2^{18.6097}$ per class	01, 08	>0.500
I2		01, 02, 08	>0.333
I3		01∼03, 08	>0.250
I4		01∼04, 08	>0.200
I5		01∼05, 08	>0.166
I6		01∼06, 08	>0.142
I7		01∼08	>0.125
I8		01∼09	>0.111
I9		01∼0A	>0.100
I10		01∼0B	>0.090
I11		01∼0C	>0.083
I12		01∼0D	>0.076
I13		01∼0E	>0.071
I14		01∼0F	>0.066

Valid accuracy is determined by the number of input differences used. For example, if three input differences are used, an accuracy higher than 0.3333 ($\frac{1}{3}$) should be achieved.

This is because it is assumed that data can be distinguished only when accuracy higher than random probability is achieved.

We perform experiments on I1∼I14 (different combinations of input differences) and achieve valid accuracies in both the number and lowercase domains (it is natural that accuracy decreases as the number of classes increases). Similar to *ModelOne*, which utilizes a single-input difference, *ModelMul* can be used as a valid distinguisher for FF1 and FF3 since it can distinguish 0x0||K differences. Tables 5 and 6 show the results of *ModelMul* according to input difference dataset for FF1 and FF3, respectively. Among I1∼I14, I2 shows the highest reliability in the number and lowercase domains (reliability means test accuracy valid accuracy). A distinguisher with high reliability has the ability to robustly classify differential characteristics. According to our results, the reliability tends to decrease as the number of differential characteristics used increases. This phenomenon is thought to occur because the more differential characteristics to be distinguished, the more complex the problem to be solved (in general, the more complex data are used, the larger the model required is). It is considered that there will be an optimal structure of the neural network according to the input difference, and this remains for our future work.

Table 5. Result of FF1 *ModelMul* according to input differences.

Dataset	Number (8 Rounds)				Lowercase (2 Rounds)			
	Training	Validation	Test	Reliability	Training	Validation	Test	Reliability
I1	0.520	0.520	0.520	0.020	0.520	0.520	0.520	0.020
I2	0.340	0.339	0.340	0.007	0.360	0.360	0.360	0.027
I3	0.260	0.260	0.260	0.010	0.270	0.270	0.270	0.020
I4	0.210	0.210	0.210	0.010	0.200	0.200	0.200	0.010
I5	0.170	0.170	0.170	0.004	0.180	0.180	0.180	0.004
I6	0.150	0.150	0.150	0.008	0.150	0.150	0.150	0.008
I7	0.130	0.130	0.130	0.005	0.130	0.130	0.130	0.005
I8	0.120	0.120	0.120	0.009	0.120	0.120	0.120	0.009
I9	0.120	0.110	0.120	0.020	0.100	0.100	0.110	0.010
I10	0.100	0.100	0.100	0.010	0.100	0.100	0.100	0.010
I11	0.090	0.090	0.090	0.007	0.090	0.090	0.090	0.007
I12	0.080	0.080	0.080	0.004	0.080	0.080	0.080	0.004
I13	0.080	0.080	0.080	0.009	0.080	0.080	0.080	0.009
I14	0.070	0.070	0.070	0.004	0.070	0.070	0.070	0.004

Table 6. Result of FF3 *ModelMul* according to input differences.

Dataset	Number (8 Rounds)				Lowercase (2 Rounds)			
	Training	Validation	Test	Reliability	Training	Validation	Test	Reliability
I1	1.00	1.00	1.00	0.500	0.55	0.55	0.55	0.050
I2	0.99	1.00	0.99	0.657	0.54	0.54	0.54	0.207
I3	0.72	0.72	0.72	0.470	0.38	0.37	0.37	0.120
I4	0.46	0.45	0.45	0.250	0.29	0.29	0.29	0.090
I5	0.33	0.33	0.33	0.164	0.24	0.23	0.23	0.064
I6	0.25	0.25	0.25	0.108	0.20	0.20	0.20	0.058
I7	0.22	0.22	0.22	0.095	0.17	0.17	0.17	0.045
I8	0.19	0.19	0.19	0.079	0.15	0.15	0.15	0.039

Table 6. *Cont.*

Dataset	Number (8 Rounds)				Lowercase (2 Rounds)			
	Training	Validation	Test	Reliability	Training	Validation	Test	Reliability
I9	0.17	0.17	0.17	0.070	0.13	0.13	0.13	0.030
I10	0.16	0.15	0.15	0.06	0.12	0.12	0.12	0.030
I11	0.14	0.14	0.14	0.057	0.11	0.11	0.11	0.027
I12	0.13	0.12	0.12	0.044	0.10	0.10	0.10	0.024
I13	0.12	0.11	0.12	0.049	0.09	0.09	0.09	0.019
I14	0.11	0.11	0.11	0.044	0.08	0.08	0.08	0.014

Figure 5 shows the reliability according to each differential characteristic and dataset. According to Dunkelman et al., the 0x08 differential characteristic was found to be the best differential in the FPE family, and 0x01 and 0x02 were found to be relatively poor differentials. *ModelOne* for FF1 shows the highest reliability in both domains when it is 0x0F. From this, we can see that 0x0F is also a good differential characteristic. *ModelOne* for FF3 shows the highest reliability in the 0x08 difference, as seen in Dunkelman et al. *ModelMul* for FF3 shows the highest reliability when it is the I2 dataset in both domains. I2 is a dataset that includes two bad differential characteristics (0x01, 0x02) and a good differential characteristic (0x08). Using this dataset is believed to have high accuracy and reliability because our distinguisher model belongs to a problem that is easy to classify.

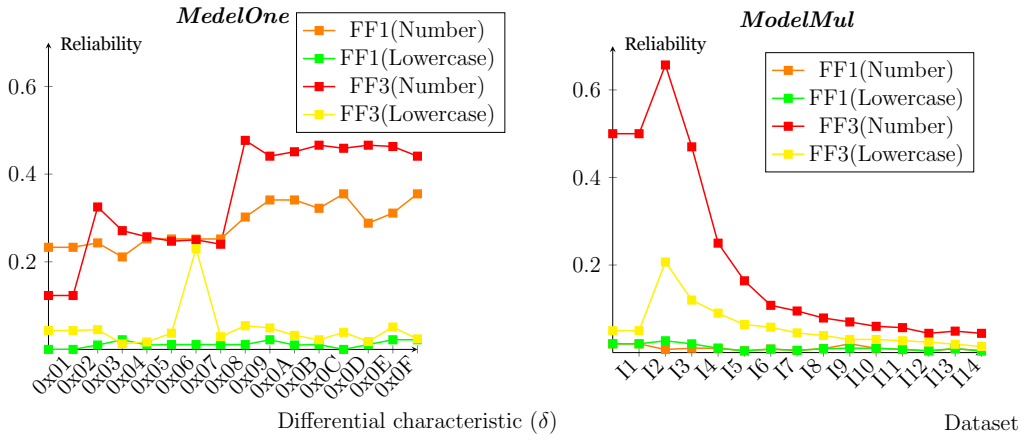

Figure 5. Reliability of *ModelOne* and *ModelMul*.

One more thing to note is that, in [26], for the classical distinguisher for FF1 and FF3, the authors use SKINNY as an inner encryption function. On the other hand, for FF1 and FF3, we used the default implementation using AES. The distinguisher attack using the input differences $0x0||K$ succeeds despite the inner encryption function being changed. Additionally, in our results, it is observed that the accuracy for 0x0F (resp. 0x08) is higher and the accuracy for 0x03 (resp. 0x01) is lower for FF1 (resp. FF3). This result seems to have come from the fact that the differential characteristic of FPE is independent of the inner encryption function. Thus, we believe that our neural distinguisher structure and differential characteristics may be applicable to other FF1 and FF3 variants as well (naturally, training needs to be performed again according to the data).

5. Conclusions

In this work, we propose the first neural distinguisher for FF1 and FF3. According to the method of classifying the input difference, the distinguisher type is divided into a binary classification model, *ModelOne*, and a multi-classification model, *ModelMul*.

In *ModelOne*, when `0x0F` (resp. `0x08`) is used, a high accuracy of 0.85 (resp. 0.98) is achieved for 10 rounds (resp. 8 rounds). In the lowercase domain, up to 2 rounds can be distinguished. In *ModelMul*, the accuracy exceeds the valid accuracy in all cases, and the highest reliability is obtained in `I2`. Through our experiments, we observe the accuracy of `0x0F` (resp. `0x08`) is higher, and the accuracy of `0x03` (resp. `0x01`) is low, relatively.

In our implementation, a different inner encryption function is used than in existing implementations, but the differential characteristic and probabilities appear to be maintained. That is, the input difference $\texttt{0x0}||K$ remains independent of the inner encryption function. Thus, it seems that our distinguisher may be utilized for variants of `FF3`.

For our future work, we plan to train our *ModelMul* on wider domains (e.g., uppercase letters, combinations of each domain). Because model optimization is important to improve generalizability, it is also important to use data with a wider range of domains. We will focus on this part. Additionally, limitations in the experimental environment made it difficult to use large amounts of data and data based on more expanded domains. We will strive to improve the experimental environment and perform more reliable validation.

Author Contributions: Software, D.K. and H.K.; Investigation, H.K. and K.J.; Writing—original draft, D.K.; Writing—review & editing, S.Y.; Supervision, H.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was financially supported by Hansung University.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Heys, H.M. A tutorial on linear and differential cryptanalysis . *Cryptologia* **2002**, *26*, 189–221. [CrossRef]
2. Taye, M.M. Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions. *Computers* **2023**, *12*, 91. [CrossRef]
3. Khaloufi, H.; Abouelmehdi, K.; Beni-Hssane, A.; Rustam, F.; Jurcut, A.D.; Lee, E.; Ashraf, I. Deep learning based early detection framework for preliminary diagnosis of COVID-19 via onboard smartphone sensors. *Sensors* **2021**, *21*, 6853. [CrossRef] [PubMed]
4. Ammer, M.A.; Aldhyani, T.H. Deep learning algorithm to predict cryptocurrency fluctuation prices: Increasing investment awareness. *Electronics* **2022**, *11*, 2349. [CrossRef]
5. Lamothe-Fernández, P.; Alaminos, D.; Lamothe-López, P.; Fernández-Gámez, M.A. Deep learning methods for modeling bitcoin price. *Mathematics* **2020**, *8*, 1245. [CrossRef]
6. Essaid, M.; Ju, H. Deep Learning-Based Community Detection Approach on Bitcoin Network. *Systems* **2022**, *10*, 203. [CrossRef]
7. Zhu, S.; Li, Q.; Zhao, J.; Zhang, C.; Zhao, G.; Li, L.; Chen, Z.; Chen, Y. A Deep-Learning-Based Method for Extracting an Arbitrary Number of Individual Power Lines from UAV-Mounted Laser Scanning Point Clouds. *Remote Sens.* **2024**, *16*, 393. [CrossRef]
8. Lata, K.; Cenkeramaddi, L.R. Deep learning for medical image cryptography: A comprehensive review. *Appl. Sci.* **2023**, *13*, 8295. [CrossRef]
9. Kim, H.; Jang, K.; Lim, S.; Kang, Y.; Kim, W.; Seo, H. Quantum Neural Network Based Distinguisher on SPECK-32/64. *Sensors* **2023**, *23*, 5683. [CrossRef] [PubMed]
10. Kim, H.; Lim, S.; Kang, Y.; Kim, W.; Kim, D.; Yoon, S.; Seo, H. Deep-learning-based cryptanalysis of lightweight block ciphers revisited. *Entropy* **2023**, *25*, 986. [CrossRef] [PubMed]
11. Gohr, A. Improving attacks on round-reduced speck32/64 using deep learning. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019 ; Springer: Berlin/Heidelberg, Germany, 2019; pp. 150–179.
12. Baksi, A. Machine learning-assisted differential distinguishers for lightweight ciphers. In *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 141–162.
13. Baksi, A.; Breier, J.; Dasu, V.A.; Hou, X.; Kim, H.; Seo, H. New Results on Machine Learning-Based Distinguishers. *IEEE Access* **2023**, *11*, 54175–54187. [CrossRef]
14. Jain, A.; Kohli, V.; Mishra, G. Deep learning based differential distinguisher for lightweight cipher PRESENT. *arXiv* **2020**, arXiv:2112.05061.
15. Rajan, R.; Roy, R.K.; Sen, D.; Mishra, G. Deep Learning-Based Differential Distinguisher for Lightweight Cipher GIFT-COFB. In *Machine Intelligence and Smart Systems: Proceedings of MISS 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 397–406.
16. Mishra, G.; Pal, S.; Krishna Murthy, S.; Prakash, I.; Kumar, A. Deep Learning-Based Differential Distinguisher for Lightweight Ciphers GIFT-64 and PRIDE. In *Machine Intelligence and Smart Systems: Proceedings of MISS 2021*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 245–257.

17. Chen, Y.; Yu, H. A New Neural Distinguisher Model Considering Derived Features from Multiple Ciphertext Pairs. *IACR Cryptol. ePrint Arch.* **2021**, *2021*, 310.
18. Benamira, A.; Gerault, D.; Peyrin, T.; Tan, Q.Q. A deeper look at machine learning-based cryptanalysis. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, 17–21 October 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 805–835.
19. Hou, Z.; Ren, J.; Chen, S. Cryptanalysis of round-reduced Simon32 based on deep learning. *Cryptol. ePrint Arch.* **2021**, *2021*, 362.
20. Yadav, T.; Kumar, M. Differential-ml distinguisher: Machine learning based generic extension for differential cryptanalysis. In Proceedings of the International Conference on Cryptology and Information Security in Latin America, Bogotá, Colombia, 6–8 October 2021; Springer: Berlin/Heidelberg, Germany, 2021; pp. 191–212.
21. Yue, X.; Wu, W. Improved Neural Differential Distinguisher Model for Lightweight Cipher Speck. *Appl. Sci.* **2023**, *13*, 6994. [CrossRef]
22. Haykin, S. *Neural Networks and Learning Machines, 3/E*; Pearson Education India: Delhi, India, 2009.
23. Stallings, W. Format-preserving encryption: Overview and NIST specification. *Cryptologia* **2017**, *41*, 137–152. [CrossRef]
24. Jang, W.; Lee, S.Y. A format-preserving encryption FF1, FF3-1 using lightweight block ciphers LEA and, SPECK. In Proceedings of the 35th Annual ACM Symposium on Applied Computing, Brno, Czech Republic, 30 March–3 April 2020; pp. 369–375.
25. Kim, H.; Kim, H.; Eum, S.; Kwon, H.; Yang, Y.; Seo, H. Parallel Implementation of PIPO and Its Application for Format Preserving Encryption. *IEEE Access* **2022**, *10*, 99963–99972. [CrossRef]
26. Dunkelman, O.; Kumar, A.; Lambooij, E.; Sanadhya, S.K. Cryptanalysis of Feistel-based format-preserving encryption. *Cryptol. ePrint Arch.* **2020**, *2020*, 1311.