

Differential-Neural Cryptanalysis of ChaCha and Related ARX Stream Ciphers

Nimai Parsa and Nitin Kumar Sharma

Indian Institute of Technology, Hyderabad, India

Abstract

The paper presents the first-ever differential neural cryptanalysis of the Latin Dances family of Salsa, ChaCha, and Forró. First, we provide a systematic differential-neural study of the ARX stream cipher ChaCha. We train and compare six neural architectures, viz: Gohr’s residual network, DBitNet, an Inception network, an MLP-ResNet, an SE-ResNet, and a self-attention network on a 3-round reduced version of ChaCha. We train in both the classical single-pair setting and a multi-pair setting that combines up to sixteen ciphertext pairs through a log-odds rule. We extend the study for breadth to other ARX ciphers, Salsa, *ChaCha, and Forró. We provide the first-ever distinguisher for 4.5 rounds of Salsa with accuracy of 0.53 for one pair and 0.60 for 16 pairs. Also, for Forró the first-ever distinguisher for 1.25 rounds of Salsa with accuracy of 1 for single and multiple pairs. We also introduce a family of experimental round-1-xor variants (*XOR-1*) that isolate the contribution of the first round’s diffusion to distinguishability. Finally, we investigate the trained 3-round ChaCha distinguisher with four explainability experiments and show that, although the output difference is its dominant feature, it is not a pure differential distinguisher: its decision is interpretable as a learned mixture of round-1 truncated differentials, five of which already account for 99% of the high-confidence ciphertext pairs.

Keywords: Neural distinguisher, Differential cryptanalysis, ChaCha, ARX ciphers, Explainability, Multi-pair distinguisher.

1 Introduction

Differential cryptanalysis [BS91] remains one of the central techniques for evaluating symmetric primitives. In classical form, a fixed input difference propagates through the round function and exploits a statistically biased output difference. The effectiveness of such an analysis depends strongly on the diffusion and nonlinear properties of the underlying round function. In a landmark work, Gohr [Goh19] replaced the hand-crafted statistical test at the end of such an analysis by a deep neural network: trained to separate ciphertext pairs produced from a chosen input difference (*real*) from random pairs (*random*), the resulting *neural distinguisher* matched and then surpassed the best classical distinguishers for round-reduced Speck32/64 and powered an improved key-recovery attack. Gohr’s introduction of neural distinguishers marked the establishment of differential-neural cryptanalysis as a widely used technique. Yet its study has concentrated on the lightweight block ciphers Speck and Simon, while the influence of the underlying neural architecture has rarely been controlled and the learned distinguishers have remained largely opaque.

A substantial line of work has since refined and generalized this idea, extending it to more than twenty symmetric primitives, to a variety of network architectures and training

Emails: nimaiparsa@gmail.com (Nimai Parsa), sharmanitinkumar685@gmail.com (Nitin Kumar Sharma)

settings, and to the question of what such a network actually learns. We review this literature in [Subsection 1.1](#).

1.1 Previous Works

Gohr [Goh19] trained a residual convolutional network on ciphertext pairs of round-reduced Speck32/64, obtained distinguishers up to 8 rounds that outperform those derived from the difference distribution table (DDT), and combined a 7-round distinguisher with a prepended differential, neutral bits, and a Bayesian key search into an 11-round attack. The field that followed is large; the survey of G rault *et al.* [GHHP26] counts 66 peer-reviewed papers on neural differential cryptanalysis across 24 primitives, of which Speck and Simon alone account for 38 and 36 experiments.

Gohr’s network remains the baseline, being used for 14 of those 24 primitives, and is later modified in major works [BBP22,DCC23,HGH⁺23,LRC23]. The main cipher-agnostic alternative is DBitNet given by Bellini *et al.* [BGHR23], whose dilated convolutions remove the need for cipher-specific reshaping. Later, Inception modules [ZWw22], MLPs [Bak22], squeeze-and-excitation networks [BGL⁺22], transformers, and UNets like architectures have also been explored [GHHP26]. But there was no conclusion on which architecture is most suitable. Baksi *et al.* [Bak22], for example, found that convolutions are unsuitable and MLPs are better on GIMLI, whereas Bellini and Rossi [BR20] and Wang *et al.* [WWH21] obtained effective convolutional distinguishers for PRESENT and Speck. Similarly, Mishra *et al.* [MPKM⁺22] preferred MLPs on GIFT and PRIDE to improve the accuracy. The survey attributes the disagreement to uncontrolled data volume, feature engineering, and training regime, and names a controlled benchmark as a central open challenge in differential neural cryptanalysis [GHHP26].

Training settings are characterized using the n - m - T - E classification introduced by Bellini *et al.* [BGHR23]. Under this classification, Gohr’s distinguisher is 2-1-CT-R, a single raw pair(2) under 1 input difference. Several pairs may be combined after scoring, through the log-odds sum formalized by Gohr *et al.* [GLN22], or processed jointly by the network [BGPT21]. Joint processing has long shown little advantage over plain aggregation [GLN22,CSYY23]. Though recent multi-pair distinguishers do reach further on Speck, Simon and Simeck [GHHP26]; Zhang *et al.* [ZWw22] warn that the number of samples must grow with the number of pairs to avoid overfitting. We use the multiple ciphertext pair rule, which keeps architecture and pair count orthogonal.

Some works also address explainability [GHHP26]. Gohr’s real-differences experiment [Goh19] masks both ciphertexts of a pair, preserving their difference but destroying the absolute values; his network stayed above chance. Benamira *et al.* [BGPT21] showed that the deeper layers encode a compressed DDT and that high-scoring pairs follow a truncated differential pattern two rounds earlier, concluding that the learned features are differential-linear. Gohr *et al.* [GLN22] provided a theoretical explanation for SPN and Feistel ciphers with a key-independent transform: a distinguisher cannot beat the DDT. Simon duly falls to exactly 0.5 under real differences, whereas Speck’s modular addition breaks that hypothesis and retains an advantage. ARX ciphers are therefore precisely where neural distinguishers exceed classical differentials [BLYZ23,LJSC24,PMC⁺22]. There is another line of work in which the identification of neutral bits affecting the distribution is analyzed [CSY23,HGH⁺23,LRC23,HLF⁺24]. Also, some other works analyze generic attribution methods and explainability techniques to interpret neural distinguishers [GLP⁺24,Seo24], but all of these works target only block ciphers, leaving the stream ciphers unattended.

Salsa20 [Ber08b] and ChaCha [Ber08a] have been analysed extensively, but classically, mainly by differential-linear cryptanalysis with probabilistic neutral bits [AFK⁺08] as the key recovery technique. These techniques are refined through multi-bit differentials [CM17], right pairs idea [BLT20], and further improvements [DGSS22,CPV⁺23,DLS25,DMSS25].

Also proposing a new cipher, Forró [CPV⁺23], as a shield against the traditional differential-linear cryptanalysis. All of these distinguishers are hand-crafted, whereas on the neural side the picture is essentially empty: the comparative review in [GHHP26] spans 24 primitives and contains no member of the Salsa/ChaCha family; AutoND [BGHR23] is cipher-agnostic but reports no results for the family. To the best of our knowledge, no dedicated differential-neural distinguisher for ChaCha has been published, no controlled comparison of architectures exists for it, and no explainability technique has been applied to a 512-bit ARX stream cipher.

1.2 Research Gaps

Despite so much progress in this domain, three important gaps still persist.

First, the body of results is dominated by the lightweight block ciphers Speck and Simon; the ARX *stream* ciphers of the Salsa/ChaCha family, widely deployed and structurally different, with a 512-bit state and a quarter-round built from modular addition, rotation, and XOR, have received comparatively very little attention.

Second, the choice of neural architecture is rarely treated as an experimental variable: most studies fix a single network and vary the cipher, leaving open how much of a distinguisher’s success is owed to the architecture rather than to the cipher’s weakness.

Third, trained distinguishers remain difficult to interpret, so it is often unclear *what* cryptographic property a high accuracy actually reflects.

1.3 Contributions

This paper addresses these three major gaps through the following contributions.

1. We provide a systematic comparison of six neural architectures, viz., Gohr’s network, DBitNet, Inception, MLP-ResNet, SE-ResNet, and a self-attention network for differential-neural cryptanalysis of a large-state ARX construction. (Section 4).
2. A case study on ChaCha at 3 rounds, evaluated in both the single-pair setting and a multi-pair setting with $m \in \{1, 2, 4, 8, 16\}$ ciphertext pairs, has been explained briefly. (Section 5).
3. We extend the same experimental framework to the related ARX stream ciphers ChaCha, *ChaCha, Salsa, and Forró and analyze their characteristics against the six different neural architectures (Section 6).
4. We also study a family of *round-1-weakened variants*, an XOR-first substitution that deliberately perturbs the first round in order to isolate its contribution to diffusion and to locate the round at which the distinguisher collapses (Section 6).
5. We provide an explainability analysis of the trained 3-round ChaCha distinguisher showing that it is not a pure differential distinguisher and that its decision is interpretable as a learned mixture of round-1 truncated differentials, which has been discussed in detail (Section 7). In particular, five discovered round-1 truncated-differential conditions collectively cover 99% of the high-confidence real pairs considered in our analysis.

Paper Organization

The rest of the paper is organized as follows. Section 2 fixes notation and recalls preliminary ideas like differential-neural distinguishers, the idea of AutoND Pipeline and DBitNet architecture, the multi-pair setting, and the evaluation metrics. Section 3 describes the target ciphers and their XOR-1 variants. Sections 4–7 present the architecture comparison,

the cross-cipher and variant results, and the explainability study as mentioned above. Section 8 discuss the possible extension of the work and Section 9 concludes and summarize this work.

2 Preliminaries Ideas

In this section, we will discuss the prerequisites for performing a differential-neural attack. First, in Subsection 2.1, we write the notations used throughout the paper. Subsection 2.2 explains the basic idea of the Differential-Neural Distinguishing attack and how we classify the real ciphertext from a random ciphertext. In Subsection 2.3, we discuss the AutoND pipeline with DBitNet as the default architecture and its extension with additional neural architectures to analyse ARX ciphers. Subsection 2.4 introduces the multi-pair neural distinguishing setting and the proposed log-odds-based evidence aggregation. Finally, Subsection 2.5 presents the evaluation metrics and statistical methodology used to assess the distinguishing performance.

2.1 Notation

We work over the binary field $\mathbb{F}_2$ and use $\oplus$ for bitwise XOR, $\boxplus$ for addition modulo 2^{32} , and $x \lll r$ for the left rotation of a 32-bit word x by r positions. A cipher state denoted by X is a vector of 32-bit words; for the Salsa/ChaCha family the state consists of 16 words (512 bits). An *input difference* Δ is a fixed XOR difference imposed on a pair of inputs, and the corresponding *output difference* of a pair (C_0, C_1) is $C_0 \oplus C_1$. r denotes the number of rounds, m the number of ciphertext pairs used by a distinguisher, and n the block or state size in bits. TD is an abbreviation for truncated differential.

2.2 Differential-Neural Distinguishing Attack

Following the methodology introduced by Gohr [Goh19], a differential-neural distinguisher is formulated as a binary classifier that receives a pair of ciphertexts and predicts whether the pair originates from the target cipher under a prescribed input difference or from a random-pair distribution. The differential-neural distinguisher for r rounds of a cipher E under input difference Δ is a neural network $\mathcal{N}$ trained as a binary classifier on labeled samples (C_0, C_1, Y) :

- For the real class $Y = 1$, a plaintext P_0 is drawn uniformly and a related plaintext is generated as $P_1 = P_0 \oplus \Delta$. Then both plaintexts are encrypted using the same secret key:

$$C_0 = E_r(P_0), \quad C_1 = E_r(P_1),$$

under a fixed (single-key) secret key; where E_r denotes the r -round construction.

- For the random class $Y = 0$, the ciphertext pair (C_0, C_1) is replaced by independently generated uniformly random data of the same dimension. The neural network therefore learns to distinguish the statistical distribution

$$(C_0, C_1 \mid Y = 1)$$

from the corresponding random distribution.

The network produces a score $N(C_0, C_1) \in (0, 1)$, which is interpreted as an estimate of $\Pr[Y = 1 \mid C_0, C_1]$.

We train the classifier using binary cross-entropy. For a balanced test set, an accuracy of 0.5 corresponds to random guessing, while an accuracy significantly above 0.5 indicates a distinguishing advantage. The network outputs a score $\mathcal{N}(C_0, C_1) \in (0, 1)$ estimating $\Pr[Y = 1]$, and is trained with binary cross-entropy. We report the validation *accuracy* together, where relevant, with the true-positive rate (TPR), the true-negative rate (TNR), and the mean-squared error (MSE); an accuracy of 0.5 corresponds to no distinguishing advantage. A distinguisher that retains accuracy above 0.5 at r rounds certifies non-random behaviour at r rounds.

2.3 The AutoND Pipeline and DBitNet

Our experiments are built on the AutoND pipeline [BGHR23], which provides a cipher-agnostic data-generation and training harness and, as its default network, DBitNet, a dilated 1D-convolutional architecture whose dilation schedule is determined by the block size so that the same network applies unchanged to ciphers of different state sizes. We retain AutoND’s training conventions (large training and validation pools, a staged round-by-round schedule) and extend the harness with the additional architectures and ciphers studied here.

An important advantage of the AutoND framework for the present study is that the same general experimental pipeline can be applied to different ARX constructions. We therefore extend the framework with the additional architectures considered in this work while keeping the underlying data-generation and evaluation procedures as consistent as possible. The architecture comparison is consequently intended to isolate the effect of the neural model rather than introducing separate training methodologies for different architectures. We have provided the details of the different architectures in [Subsection 4.1](#).

2.4 The Multi-Pair Setting

A single ciphertext pair can carry only limited evidence at higher rounds. Multi-pair neural distinguishing has previously been investigated by Chen *et al.* [CSY23]. In the *multi-pair* setting, a sample consists of m ciphertext pairs produced under the same secret key from the same input difference. A multi-pair sample consists of m ciphertext pairs $(C_0^{(i)}, C_1^{(i)})$ for $1 \leq i \leq m$, generated under the same secret key and using the same input difference. Each pair is independently evaluated by the trained single-pair neural distinguisher. We score each pair independently with $\mathcal{N}$ and combine the scores through a log-odds (logit) sum,

$$\ell = \sum_{i=1}^m \log \frac{p_i}{1 - p_i}, \quad p_i = \mathcal{N}(C_0^{(i)}, C_1^{(i)}), \quad (1)$$

The resulting multi-pair score is $P_{\text{multi}} = \sigma(\ell)$, where σ denotes the logistic function. The sample is classified as real when $P_{\text{multi}} \geq 0.5$. This rule is the optimal combination under conditional independence of the per-pair scores and converts a weak per-pair advantage into a much stronger per-sample one. Throughout the work, we evaluate $m \in \{1, 2, 4, 8, 16\}$ pairs.

2.5 Evaluation Metrics and Statistical Interpretation

For a balanced binary classification problem, random guessing corresponds to an accuracy of 0.5. We therefore use validation accuracy as the primary measure of distinguishing performance. However, accuracy alone does not fully characterize the quality of a neural distinguisher, particularly when several ciphertext pairs are aggregated.

For a classifier producing a score $p \in (0, 1)$, we additionally consider the true-positive rate (TPR), true-negative rate (TNR), and mean-squared error (MSE). The TPR measures

the fraction of real ciphertext pairs correctly identified as real, whereas the TNR measures the fraction of random pairs correctly rejected.

For multi-pair experiments, we apply the same trained single-pair distinguisher independently to each ciphertext pair and combine the resulting scores using the log-odds rule described in the previous section. This design allows us to distinguish improvements caused by the neural architecture from improvements caused solely by the availability of additional ciphertext pairs. Unless otherwise stated, all reported accuracies are computed on data that are independent of the training and validation sets.

3 Target Ciphers

In this section, we describe the design and structure of ChaCha and the other ARX stream ciphers like Salsa, *ChaCha, and Forró considered in this work. We provide detailed descriptions of their respective variants and components to clearly distinguish the constructions and establish the scope of our differential neural cryptanalysis.

3.1 Salsa

Salsa20 [Ber08b] was designed by D. J. Bernstein in 2005 and later appeared in the eSTREAM project in 2008. Salsa20 operates on a 512-bit state arranged as a 4×4 matrix of 32-bit words: four constants (c_0, c_1, c_2, c_3), eight key words ($k_0, \dots, k_7$), two nonce (v_0, v_1) and two counter (t_0, t_1) words. The cipher uses 32-bit addition, 32-bit rotation, and exclusive OR operations in the state mixing function, popularly known as the quarter-round function. The constants values are $c_0 = 0x61707865, c_1 = 0x3320646e, c_2 = 0x79622d32$ and $c_3 = 0x6b206574$.

$$X = \begin{array}{|c|c|c|c|} \hline x_0 = c_0 & x_1 = k_0 & x_2 = k_1 & x_3 = k_2 \\ \hline x_4 = k_3 & x_5 = c_1 & x_6 = v_0 & x_7 = v_1 \\ \hline x_8 = t_0 & x_9 = t_1 & x_{10} = c_2 & x_{11} = k_4 \\ \hline x_{12} = k_5 & x_{13} = k_6 & x_{14} = k_7 & x_{15} = c_3 \\ \hline \end{array}$$

The state is mixed by a quarter-round that updates four words (a, b, c, d) through the ARX sequence

$$\begin{aligned} b \oplus &= (a \boxplus d) \lll 7; & c \oplus &= (b \boxplus a) \lll 9; \\ d \oplus &= (c \boxplus b) \lll 13; & a \oplus &= (d \boxplus c) \lll 18; \end{aligned}$$

The quarter-round function applies to the matrix elements in two ways, one along the column and the other along the row. These are applied alternatively 10 times, completing the 20 rounds of Salsa20. Along the four column the four words (a, b, c, d) are operated in the order (x_0, x_4, x_8, x_{12}) , $(x_5, x_9, x_{13}, x_{17})$, $(x_{10}, x_{14}, x_{18}, x_{22})$ and $(x_{15}, x_{19}, x_{23}, x_{27})$ respectively. Similarly, along the four rows the order is (x_0, x_1, x_2, x_3) , (x_5, x_6, x_7, x_4) , $(x_{10}, x_{11}, x_8, x_9)$ and $(x_{15}, x_{12}, x_{13}, x_{14})$ respectively.

At the end of n rounds, there is also a key stream generation step, in which the initial state X is added to the state obtained after n rounds. The key stream block Z of 16 words or 512 bits is obtained as $Z = X + X^{(n)}$.

3.2 ChaCha

ChaCha [Ber08a] operates on a 512-bit state arranged as a 4×4 matrix of 32-bit words: a row of constants, eight key words, and four counter/nonce words.

$$X = \begin{array}{|c|c|c|c|} \hline x_0 = c_0 & x_1 = c_1 & x_2 = c_2 & x_3 = c_3 \\ \hline x_4 = k_0 & x_5 = k_1 & x_6 = k_2 & x_7 = k_3 \\ \hline x_8 = k_4 & x_9 = k_5 & x_{10} = k_6 & x_{11} = k_7 \\ \hline x_{12} = t_0 & x_{13} = t_1 & x_{14} = v_0 & x_{15} = v_1 \\ \hline \end{array}$$

The state is mixed by a quarter-round that updates four words (a, b, c, d) through the ARX sequence

$$\begin{array}{ll} a \boxplus= b, & d \leftarrow (d \oplus a) \lll 16; & c \boxplus= d, & b \leftarrow (b \oplus c) \lll 12; \\ a \boxplus= b, & d \leftarrow (d \oplus a) \lll 8; & c \boxplus= d, & b \leftarrow (b \oplus c) \lll 7, \end{array}$$

The quarter-round function applies to the matrix elements in two ways, one along the column and the other along the diagonals. These are applied alternatively 10 times, completing the 20 rounds of ChaCha. Along the four column the four words (a, b, c, d) are operated in the order (x_0, x_4, x_8, x_{12}) , (x_1, x_5, x_9, x_{13}) , $(x_2, x_6, x_{10}, x_{14})$ and $(x_3, x_7, x_{11}, x_{15})$ respectively. Similarly, along the four diagonals the order is $(x_0, x_5, x_{10}, x_{15})$, $(x_1, x_6, x_{11}, x_{12})$, (x_2, x_7, x_8, x_{13}) and (x_3, x_4, x_9, x_{14}) respectively.

At the end of n rounds, there is also a key stream generation step, in which the initial state X is added to the state obtained after n rounds. The key stream block Z of 16 words or 512 bits is obtained as $Z = X + X^{(n)}$.

3.3 Forró

Forró [CPV⁺23] is a more recent ARX stream cipher whose *five*-input quarter-round, with rotations (10, 27, 8), achieves stronger diffusion per operation and therefore reaches full mixing in fewer rounds; Both ciphers are driven through the same AutoND harness and the same architecture suite as ChaCha.

$$X = \begin{array}{|c|c|c|c|} \hline x_0 = k_0 & x_1 = k_1 & x_2 = k_2 & x_3 = k_3 \\ \hline x_4 = t_0 & x_5 = t_1 & x_6 = c_0 & x_7 = c_1 \\ \hline x_8 = k_4 & x_9 = k_5 & x_{10} = k_6 & x_{11} = k_7 \\ \hline x_{12} = v_0 & x_{13} = v_1 & x_{14} = c_2 & x_{15} = c_3 \\ \hline \end{array}$$

The state is mixed by a quarter-round that updates four words (a, b, c, d, e) through the ARX sequence

$$\begin{array}{lll} d \boxplus= e, & c \oplus= d, & b = (b \boxplus c) \lll 10; \\ a \boxplus= b, & e \oplus= a, & d = (d \boxplus e) \lll 27; \\ c \boxplus= d, & b \oplus= c, & a = (a \boxplus b) \lll 8. \end{array}$$

The quarter-round function applies to the matrix elements in two ways, one along the column and the other along the row. These are applied alternatively 7 times, completing the 14 rounds of Forró. Along the four column the four words (a, b, c, d, e) are operated in the order $(x_0, x_4, x_8, x_{12}, x_3)$, $(x_1, x_5, x_9, x_{13}, x_0)$, $(x_2, x_6, x_{10}, x_{14}, x_1)$ and $(x_3, x_7, x_{11}, x_{15}, x_2)$ respectively. Similarly, along the four diagonals the order is $(x_0, x_5, x_{10}, x_{15}, x_3)$, $(x_1, x_6, x_{11}, x_{12}, x_0)$, $(x_2, x_7, x_8, x_{13}, x_1)$ and $(x_3, x_4, x_9, x_{14}, x_2)$ respectively.

At the end of n rounds, there is also a key stream generation step, in which the initial state X is added to the state obtained after n rounds. The key stream block Z of 16 words or 512 bits is obtained as $Z = X + X^{(n)}$.

3.4 *ChaCha

*ChaCha was introduced by Coutinho *et al.* [CPdSJB20] in 2020 by analyzing which rotation constants provide the most diffusion in the standard ChaCha quarter round. The cipher operates on a 512-bit state arranged as a 4×4 matrix of 32-bit words: a row of constants, eight key words, and four counter/nonce words.

$$X = \begin{array}{|c|c|c|c|} \hline x_0 = c_0 & x_1 = c_1 & x_2 = c_2 & x_3 = c_3 \\ \hline x_4 = k_0 & x_5 = k_1 & x_6 = k_2 & x_7 = k_3 \\ \hline x_8 = k_4 & x_9 = k_5 & x_{10} = k_6 & x_{11} = k_7 \\ \hline x_{12} = t_0 & x_{13} = t_1 & x_{14} = v_0 & x_{15} = v_1 \\ \hline \end{array}$$

The structure of *ChaCha is the same as the standard ChaCha cipher, just the rotation constants (16, 12, 8, 7) are replaced with (19, 17, 25, 11). The state is mixed by a quarter-round that updates four words (a, b, c, d) through the ARX sequence

$$\begin{aligned} a \boxplus b, d &\leftarrow (d \oplus a) \lll 19; & c \boxplus d, b &\leftarrow (b \oplus c) \lll 17; \\ a \boxplus b, d &\leftarrow (d \oplus a) \lll 25; & c \boxplus d, b &\leftarrow (b \oplus c) \lll 11, \end{aligned}$$

The quarter-round function applies to the matrix elements in two ways, one along the column and the other along the diagonals. These are applied alternatively 10 times, completing the 20 rounds of ChaCha. Along the four column the four words (a, b, c, d) are operated in the order (x_0, x_4, x_8, x_{12}) , (x_1, x_5, x_9, x_{13}) , $(x_2, x_6, x_{10}, x_{14})$ and $(x_3, x_7, x_{11}, x_{15})$ respectively. Similarly, along the four diagonals the order is $(x_0, x_5, x_{10}, x_{15})$, $(x_1, x_6, x_{11}, x_{12})$, (x_2, x_7, x_8, x_{13}) and (x_3, x_4, x_9, x_{14}) respectively.

At the end of n rounds, there is also a key stream generation step, in which the initial state X is added to the state obtained after n rounds. The key stream block Z of 16 words or 512 bits is obtained as $Z = X + X^{(n)}$.

We will see in Section 6 that the change in rotation constants leads to a difference in the differential neural cryptanalysis.

3.5 XOR-1 Round Variant

In this part, we discuss the tweaked versions of the ARX ciphers. To study the contribution of first-round nonlinear diffusion, we construct controlled variants in which every modular addition operation $\boxplus$ in the first round is replaced by XOR, and all subsequent rounds remain unchanged. Because XOR is $\mathbb{F}_2$ -linear, this weakens the diffusion contributed by the first round and pushes the nonlinear mixing one round later, so a distinguisher must reach further to collapse. We denote the resulting variants by Salsa_{XOR}, ChaCha_{XOR}, *ChaCha_{XOR} and Forró_{XOR}. The motivation for these variants comes from the idea of the right pair introduced by Beierle *et al.* [BLT20], where they discuss how to control the diffusion in the first round of ARX ciphers.

- **Salsa_{XOR}**: A variant of Salsa in which the 16 modular additions ($\boxplus$) in the first round are replaced by 16 bitwise XOR operations ($\oplus$), while all subsequent rounds remain unchanged.
- **ChaCha_{XOR}**: A variant of ChaCha obtained by replacing the 16 modular additions ($\boxplus$) in the first round with 16 bitwise XOR operations ($\oplus$), leaving the remaining rounds unchanged.
- ***ChaCha_{XOR}**: A variant of *ChaCha in which the first-round 16 modular additions ($\boxplus$) are replaced by 16 XOR operations ($\oplus$), preserving the modifications of *ChaCha in the subsequent rounds.

- **Forró_{XOR}**: A variant of Forró where the 24 modular additions ($\boxplus$) in the first round are replaced with 24 XOR operations ($\oplus$), while retaining the original operations in the remaining rounds.

The perturbations are not cipher variants proposed as cipher designs; they are instruments for measuring the marginal contribution of the first round to distinguishability.

4 Architecture Comparison

The choice of neural architecture is an important experimental variable in neural cryptanalysis, as different architectures impose different inductive biases on the representation and processing of the input. ChaCha, as a widely studied ARX-based stream cipher, provides an interesting setting for investigating how these architectural choices influence neural distinguishing performance. In this work, we analyze a range of neural architectures to examine how different mechanisms for processing the input affect the ability of the distinguisher to capture cryptographic properties. This comparison allows us to assess whether the observed distinguishing capability is dependent on a particular architectural bias and to identify architectures that are better suited to capturing the underlying differential properties of ChaCha.

Our objective is therefore not simply to identify the architecture with the highest accuracy. Instead, we compare architectures with substantially different inductive biases under a common training and data-generation procedure. This allows us to investigate whether the distinguishing signal is associated with local word structure, multi-scale dependencies, channel interactions, or global relationships between state positions.

4.1 The Six Architectures

In this section, we compare six neural architectures spanning the main inductive biases used in the literature. [Table 1](#) summarizes the strengths and weaknesses of these architectures.

- **Gohr’s network** [Goh19] is the original bit-sliced residual convolutional network: small, fast, and strongly biased toward local word structure, but with a limited receptive field. Its convolutional structure imposes a locality bias, allowing the network to learn relationships among nearby input positions while residual connections facilitate optimization of deeper models.
- **DBitNet** [BGHR23] is the AutoND default; its geometrically dilated wide narrow blocks separate long-range and local mixing, though its receptive field is still bounded by the input size. This property is potentially useful for ARX constructions, where rotations and word-level operations create dependencies across multiple bit positions.
- **Inception** (INCNet) uses parallel multi-scale convolutions to capture local and mid-range structure at the cost of more parameters. In the present setting, such multi-scale processing may be useful because differential information in an ARX cipher can appear at several spatial scales as bit dependencies propagate through successive modular additions and rotations.
- **MLP-ResNet** is a dense skip-connected network that makes no spatial assumption; it is fully general but ignores the word layout and is harder to optimize. This makes it particularly useful as a control architecture: if its performance differs substantially from that of convolutional models under the same training conditions, the difference provides evidence that the organization of the state contributes to the learnable distinguishing signal.

- **SE-ResNet** augments a residual convolutional trunk with squeeze-and-excitation channel attention that re-weights feature channels. We therefore include SE-ResNet to determine whether adaptive feature re-weighting improves over conventional convolutional processing.
- **Attention** applies self-attention over the ciphertext words, giving a global receptive field able to model long-range bit dependencies, but is heavier and more data-hungry. This makes it an appropriate candidate for large-state ARX constructions in which relevant differential dependencies may span multiple words. Its principal disadvantage is increased computational and data requirements.

Table 1: Architectures evaluated: inductive biases.

Architecture	Strengths	Weaknesses
Gohr	Small, fast; strong locality bias	Limited receptive field
DBitNet	Adaptive dilation; separates long-range and local mixing	Receptive field bounded by input size
Inception	Multi-scale local + mid-range features	Heavier; more parameters
MLP-ResNet	No spatial assumption; fully general	Ignores word structure; hard to optimize
SE-ResNet	Channel attention re-weights features	Ignores spatial structure
Attention	Global receptive field; long-range mixing	Heavier; data-hungry

4.2 Experimental Setup

All distinguishers are trained using the same general data-generation and evaluation pipeline. Unless otherwise stated, the input difference is fixed to the selected single-bit difference located at word 13, bit 15.

For each round under consideration, we generate 10^7 training samples and 10^6 validation samples. The batch size is 5000, and training proceeds in a round-by-round schedule that advances to round $r+1$ from the weights learned at round r until the validation accuracy saturates near 0.5. For the multi-pair settings, the per-pair scores of a trained network are combined as in Eq. (1).

To ensure a fair comparison, the same input difference, class balance, training-data size, validation-data size, and evaluation procedure are used for the different architectures. Details of the optimizer, learning rate, implementation environment, hardware, and random seeds are provided to facilitate reproducibility.

5 ChaCha Case Study

In this section, we investigate ChaCha in detail because it provides the principal large-state ARX test case in our study. We examine the distinguishing performance of all six architectures at reduced round counts and then investigate the effect of combining multiple ciphertext pairs. The experiments are designed to answer whether different neural architectures learn substantially different amounts of information from the same differential distribution. We work in the single-key scenario and, unless stated otherwise, place the input difference in word 13, bit 15 (written `0x80000000000000000000`), the single-bit difference selected by the input-difference search.

Table 2 reports the validation accuracy of the six architectures on 3-round ChaCha for $m \in \{1, 2, 4, 8, 16\}$ pairs. The Gohr single-pair accuracy at three rounds is the previously reported baseline of Gohr, Leander, and Neumann [GLN22]; the multi-pair columns and the remaining architectures are evaluated here for the first time. At 3.5 and 4 rounds, every architecture collapses to 0.5 regardless of m ; we therefore omit those columns and discuss them in the text.

Table 2: ChaCha, 3 rounds: accuracy by architecture and number of ciphertext pairs m . All entries are produced in this work except the Gohr single-pair value (†), which is the previously published baseline of [GLN22].

Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
Gohr	0.9890 †	0.9992	1.0000	1.0000	1.0000
DBitNet	0.9888	0.9991	1.0000	1.0000	1.0000
Inception	0.9876	0.9990	1.0000	1.0000	1.0000
SE-ResNet	0.8632	0.9291	0.9786	0.9973	0.9999
Attention	0.9850	0.9985	1.0000	1.0000	1.0000
MLP-ResNet	0.5015	0.5015	0.5015	0.5015	0.5015

From Table 2 we draw three important observations:

- First, *three rounds is a sharp wall*: five of the six architectures distinguish 3-round ChaCha with single-pair accuracy near 0.99, but every architecture, including the strongest, drops to chance at 3.5 and 4 rounds.
- Second, *word structure matters*: the convolutional and attention models cluster around 0.99, whereas the structure-agnostic MLP-ResNet stays at chance (0.5015), showing that the signal is carried by the spatial arrangement of the 32-bit words rather than by any global function of the raw bits. SE-ResNet is the weakest of the structured models at $m=1$ (0.8632), indicating that channel re-weighting alone is a poorer prior here than explicit local convolution.
- Third, *the multi-pair rule is decisive*: combining as few as four pairs lifts all of the structured models to 1.0000, and even SE-ResNet reaches 0.9999 by $m=16$. For ChaCha at 3 rounds, we therefore recommend Gohr’s network or DBitNet as the default, with the multi-pair rule applied whenever several pairs under a common key are available.

6 Results Across Ciphers and their XOR-1 Variants

In this section, we present the distinguishing accuracy results obtained for the Latin Dance family of ciphers and their corresponding XOR-1 variants. In Subsection 6.1, we present our results for the Salsa cipher, where the distinguisher achieves an accuracy of 0.60 for up to 4.5 rounds. In Subsection 6.2, we compare our results with those reported in previous work on 3-round ChaCha. Our approach significantly improves the distinguishing accuracy from 0.989 to 1.0 for 3-round ChaCha. In Subsection 6.3, we present the results obtained for the Forró cipher, for which we achieve distinguishing results up to 1.25 rounds.

Furthermore, in Subsection 6.4, we present the results for *ChaCha, a modified variant of the ChaCha cipher. Finally, to investigate the diffusion behavior of the first-round function in the aforementioned ciphers, we evaluate their corresponding XOR-1 variants and present the resulting accuracy measurements in Subsection 6.5. These results provide additional insight into the role of the first-round diffusion function in the distinguishing capability of the neural models.

6.1 Salsa

The Salsa results demonstrate a substantially different distinguishing profile from ChaCha under the same general experimental framework. The structured architectures remain highly distinguishable at 3 and 3.5 rounds, while DBitNet retains a measurable advantage at 4 rounds and a smaller but non-negligible advantage at 4.5 rounds. DBitNet distinguishes 4-round Salsa with a clear advantage (0.6187 at $m=1$, rising to 0.8751 at $m=16$) and retains a measurable advantage at 4.5 rounds ($0.5325 \rightarrow 0.6044$), placing the practical wall for Salsa half a round.

Table 3: Salsa: accuracy by number of pairs m . Upper block: all architectures coincide at 3 and 3.5 rounds (representative values shown). Lower block: DBitNet at 4 and 4.5 rounds.

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
3	all (struct.)	1.0000	1.0000	1.0000	1.0000	1.0000
3.5	all (struct.)	1.0000	1.0000	1.0000	1.0000	1.0000
4	DBitNet	0.6187	0.6598	0.7197	0.7937	0.8751
4.5	DBitNet	0.5325	0.5462	0.5622	0.5833	0.6044
5	all	0.5000	0.5000	0.5000	0.5000	0.5000

6.2 ChaCha

In this section, we compare our results for the 3-round ChaCha distinguisher with those reported in previous neural cryptanalysis work [GLN22]. The comparison considers not only the distinguishing accuracy but also the neural-network architecture and the corresponding m -value, which represents the number of ciphertext samples used as input to the distinguisher. These parameters are important for assessing the effectiveness and practical requirements of the proposed approach.

Table 4: Comparison of differential-neural distinguishing results for 3-round ChaCha

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
3	Gohr’s residual network [GLN22]	0.989	0.999	1.0000	1.0000	1.0000
	all (struct.)	1.0000	1.0000	1.0000	1.0000	1.0000
3.25	all	0.5010	0.5010	0.5010	0.5010	0.5010

6.3 Forró

In this section, we present the first-ever differential neural cryptanalysis of the Forró cipher. Forró’s five-input quarter-round diffuses much faster, and the distinguishers saturate already at one round.

Table 5: Forró: accuracy by number of pairs m (all six architectures).

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
1	all	1.0000	1.0000	1.0000	1.0000	1.0000
1.25	all (excl. MLP)	1.0000	1.0000	1.0000	1.0000	1.0000
	MLP-ResNet	0.9999	1.0000	1.0000	1.0000	1.0000
1.5	all	0.5009	0.5009	0.5009	0.5009	0.5009

We use the input difference `0x3000000000000000` to find the differential neural distinguisher. Table 5 reports 1 and 1.25 rounds, where every architecture achieves accuracy 1.0000 apart from a single 0.9999 entry for MLP-ResNet at 1.25 rounds, $m=1$. The contrast with ChaCha and Salsa quantifies how much more diffusion Forró packs into each round.

6.4 *ChaCha

In this section, we investigate the accuracy values for *ChaCha, the modified version of ChaCha. In this version, the state representation and quarter-round structure remain unchanged from ChaCha, while the standard rotation constants(16, 12, 8, 7) are replaced by(19, 17, 25, 11). We observe that the rotation constants affect the neural distinguishability, and hence we can only reach up to 2.5 rounds with accuracy greater than 0.5.

Table 6: *ChaCha: accuracy by number of pairs m (all six architectures).

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
2	all	1.0000	1.0000	1.0000	1.0000	1.0000
2.25	Gohr	0.9989	0.9999	1.0000	1.0000	1.0000
	DBitNet	0.9985	0.9999	1.0000	1.0000	1.0000
	Inception	0.9989	0.9999	1.0000	1.0000	1.0000
	SE-ResNet	0.9989	0.9999	1.0000	1.0000	1.0000
	Attention	0.9986	0.9999	1.0000	1.0000	1.0000
	MLP-ResNet	0.9976	0.9999	1.0000	1.0000	1.0000
2.5	Gohr	0.9989	0.9999	1.0000	1.0000	1.0000
	DBitNet	0.9985	0.9999	1.0000	1.0000	1.0000
	Inception	0.9989	0.9999	1.0000	1.0000	1.0000
	SE-ResNet	0.9989	0.9999	1.0000	1.0000	1.0000
	Attention	0.9986	0.9999	1.0000	1.0000	1.0000
	MLP-ResNet	0.9976	0.9999	1.0000	1.0000	1.0000
2.75	all	0.5006	0.5006	0.5006	0.5006	0.5006

6.5 XOR-1 Variants

Replacing the first-round addition by XOR We now turn to the controlled XOR-1 variants of the ciphers mentioned in Subsection 3.5. The XOR-1 experiments provide a controlled measurement of the role of first-round nonlinear operations. Replacing modular additions with XOR increases the number of rounds for which a distinguishing signal remains observable.

Table 7: Salsa_{XOR} at 4 and 4.5 rounds.

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
4	Gohr	0.6803	0.7430	0.8203	0.8994	0.9613
	DBitNet	0.6757	0.7380	0.8312	0.8925	0.9563
	Inception	0.6807	0.7429	0.8189	0.8978	0.9597
	Attention	0.6764	0.7374	0.8129	0.8921	0.9556
4.5	Gohr	0.5903	0.6273	0.6768	0.7410	0.8193
	DBitNet	0.5884	0.6241	0.6725	0.7362	0.8124
	Inception	0.5885	0.6252	0.6752	0.7385	0.8141
	Attention	0.5890	0.6260	0.6754	0.7392	0.8164
5	Attention	0.5000	0.5000	0.5000	0.5000	0.5000

6.5.1 Salsa_{XOR}

The same effect appears for Salsa, whose XOR-1 variant is distinguishable at 4 and 4.5 rounds across all structured architectures (Table 7).

6.5.2 ChaCha_{XOR}

In the context of ChaCha, linearizing the first round pushes the wall to a later round, exactly as expected. The XOR-1 variant remains distinguishable at 3.25 rounds, where standard ChaCha is already at chance: DBitNet reaches 0.5908 at $m=1$ and 0.8299 at $m=16$, while the attention network gives a weaker $0.5442 \rightarrow 0.6510$ (Table 8).

Table 8: ChaCha_{XOR}, 3.25 rounds.

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
3	all	1.0000	1.0000	1.0000	1.0000	1.0000
3.25	DBitNet	0.5908	0.6299	0.6833	0.7512	0.8299
	Attention	0.5442	0.5569	0.5784	0.6090	0.6510
3.5	Attention	0.5000	0.5000	0.5000	0.5000	0.5000

6.5.3 Forró_{XOR}

In the case of Forró, the XOR-1 variant pushes strong signal to 1.5 and 2 rounds (Table 9). In every case, the gain from linearizing one round is roughly a quarter to a full round of additional reach, a direct measure of how much nonlinear diffusion the first round normally supplies.

Table 9: Forró_{XOR} at 1.5 and 2 rounds.

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
1.5	Gohr	1.0000	1.0000	1.0000	1.0000	1.0000
	DBitNet	0.9999	1.0000	1.0000	1.0000	1.0000
2	Gohr	0.9998	1.0000	1.0000	1.0000	1.0000
	DBitNet	0.9984	0.9999	1.0000	1.0000	1.0000
	MLP-ResNet	0.9821	0.9982	1.0000	1.0000	1.0000
2.5	Attention	0.5000	0.5000	0.5000	0.5000	0.5000

6.5.4 *ChaCha_{XOR}

For *ChaCha, Table 6 shows accuracy of 1.0000 at 2 rounds for all architectures and near-0.999 single-pair accuracy at 2.25 and 2.5 rounds, saturating to 1.0000 by $m=4$. Combining the two perturbations, *ChaCha_{XOR} remains at accuracy 1.0000 (all architectures) at 2.25 rounds and at ≥ 0.9995 single-pair, 1.0000 multi-pair at 2.5 rounds (Table 10). The rotation constants thus measurably shift the boundary, confirming that the choice of rotation amounts in the ARX skeleton governs the rate at which a single-bit difference becomes indistinguishable.

Table 10: *ChaCha_{XOR} at 2.25 and 2.5 rounds.

Rounds	Architecture	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
2.25	all (excl. MLP)	1.0000	1.0000	1.0000	1.0000	1.0000
	MLP-ResNet	0.9997	1.0000	1.0000	1.0000	1.0000
2.5	Gohr/DBitNet	0.9997	1.0000	1.0000	1.0000	1.0000
	MLP-ResNet	0.9936	0.9997	1.0000	1.0000	1.0000
2.75	All	0.5007	0.5007	0.5007	0.5007	0.5007

Conclusion: These results mentioned in this subsection support the hypothesis that first-round modular additions contribute substantially to the destruction of differential structure. Since XOR does not introduce carry-dependent nonlinearities, replacing addition with XOR reduces the nonlinear mixing introduced during the first round and allows differential information to survive further into the computation.

7 Explainability of the 3-Round ChaCha Distinguisher

In this attack against ChaCha, our primary focus was on improving the accuracy of ciphertext distinguishing. However, achieving a high distinguishing accuracy alone is not sufficient from a cryptanalytic perspective; it is equally important to understand which input differences contribute to the observed distinguishing capability and why. Hence, in this section, we analyze the trained 3-round ChaCha DBitNet distinguisher through four progressively more structured experiments. First, the real-differences experiment tests whether the classifier requires absolute ciphertext values in addition to the output difference. Second, output-difference-only training measures the distinguishing information contained in the XOR difference alone. Third, bit sampling investigates whether the full 512-bit state is required. Finally, we perform ciphertext-pair analysis via Truncated Differential structures that explain the classifier’s decisions.

7.1 Real-Differences Experiment

In this part, we apply Gohr’s real-differences test [Goh19] on 3-round ChaCha. In the real-differences experiment, for half of the samples, both ciphertexts of a real pair are XORed by a common uniformly random mask M , mapping $(C_0, C_1) \mapsto (C_0 \oplus M, C_1 \oplus M)$; this preserves the output difference $C_0 \oplus C_1$ but randomizes the absolute ciphertext values. A pure differential distinguisher must collapse to 0.5 on this task. Instead, the network attains accuracy 0.9502 (TPR 0.9881, TNR 0.9124, MSE 0.0418), so it exploits information in the absolute ciphertext values beyond the output difference alone: it is *not* a pure differential distinguisher. This result does not imply that absolute ciphertext values are the dominant source of information. To quantify their relative contribution, we perform output-difference-only training in the next experiment.

7.2 Output-Difference-Only Training

The output-difference-only experiment provides a complementary measurement in comparison to the real-differences experiment. In this, a new distinguisher is trained using only $\Delta C = C_0 \oplus C_1$ as input, with no access to the ciphertext words. This distinguisher reaches validation accuracy 0.9703, close to the 0.9888–0.9890 of the full distinguisher. The output difference is therefore the *dominant* feature; the absolute ciphertext context contributes the remaining few percent. Combining the two experiments therefore leads to a more precise interpretation that a distinguisher is built primarily on the output difference and refined by ciphertext context.

7.3 Bit-Sampling Importance

The next task is to find how many ciphertext bits the distinguisher actually needs, by retraining DBitNet with only a fixed subset of the 512 input bits visible (the bits selected from a per-bit sensitivity ranking [CSY23]).

Table 11: Accuracy of DBitNet on 3-round ChaCha with only a subset of input bits visible.

Bits (of 512)	$m=1$	$m=2$	$m=4$	$m=8$	$m=16$
100	0.9494	0.9886	0.9992	1.0000	1.0000
58	0.9516	0.9891	0.9993	1.0000	1.0000
25	0.8501	0.9296	0.9811	0.9982	1.0000
10	0.6449	0.6968	0.7625	0.8420	0.9188

Table 11 shows that 58 bits already recover the full performance essentially and that even 25 bits reach $\approx 100\%$ accuracy once 16 pairs are combined; performance degrades only when the budget falls to 10 bits. Most of the 512 output bits are thus redundant for the distinguishing task.

This result indicates that the learned distinguisher does not require all 512 state bits. It also illustrates an important interaction between feature selection and multi-pair aggregation: additional ciphertext pairs can compensate for the loss of information caused by restricting the visible feature set. Figure 1 shows the underlying per-bit sensitivity profile.

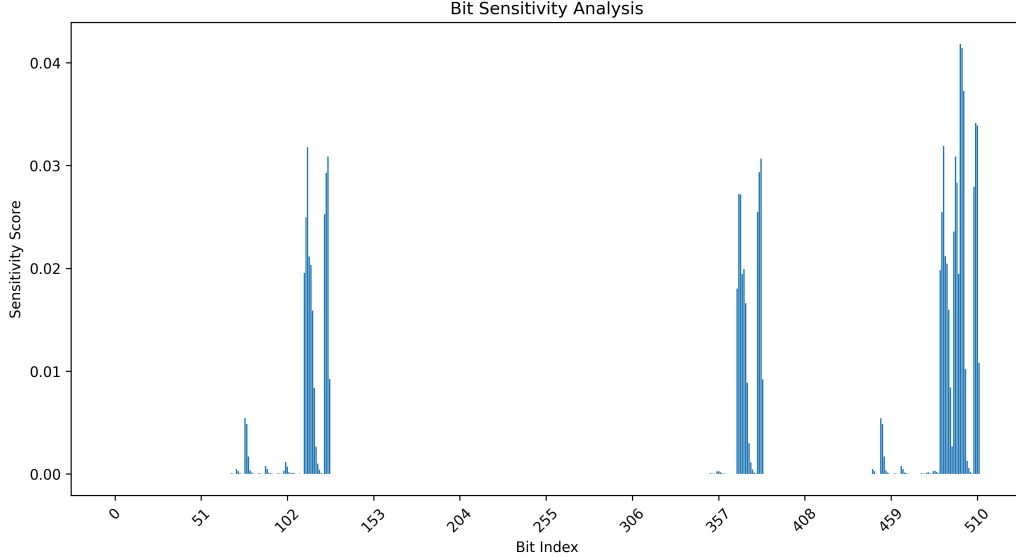


Figure 1: Per-bit sensitivity of the 3-round ChaCha DBitNet distinguisher. A small minority of bits carry most of the distinguishing signal.

7.4 Ciphertext-Pair Analysis via Truncated Differentials

To identify the cryptographic structure associated with the classifier’s decisions, we examine the XOR difference at intermediate states. Let $S_r^{(0)}$ and $S_r^{(1)}$ denote the two states after round r . Their difference is

$$\Delta S_r = S_r^{(0)} \oplus S_r^{(1)},$$

with $b_j = \Pr[\Delta S_{r,j} = 1] - \frac{1}{2}$. A bit is regarded as constrained when $|b_j| > \tau$, where τ is the selected bias threshold. The corresponding set of constrained positions defines a truncated-differential mask.

This representation is useful because it does not require the exact value of the complete state difference. Instead, it records only those bit positions for which the difference exhibits a statistically significant departure from an unbiased distribution.

To perform this experiment, we adopted the good/bad-pair analysis of Benamira *et al.* [BGPT21] for ChaCha. Scoring 100,000 real pairs with the trained network, we split them into a high-confidence set G (score ≥ 0.9) and a low-confidence set B (score ≤ 0.1). The network is extremely confident: $|G| = 96,372$ (96.4%) and $|B| = 419$ (0.4%), with a true-positive rate of 0.988 on real pairs. Almost every real pair is recognized with high confidence. Now, we discuss the per-bit output XOR-difference for both high confidence set G and low confidence set B based on the signal.

The signal is not at the output The per-bit bias of $C_0 \oplus C_1$ over G is essentially flat: the maximum absolute bias is ≈ 0.10 (a single bit just above 0.1, none above 0.3), and over B it is pure noise (≤ 0.092). Three rounds of ChaCha diffuse the input difference across the whole 512-bit state, so the network cannot be reading a simple per-bit output statistic as shown in Figure 2.

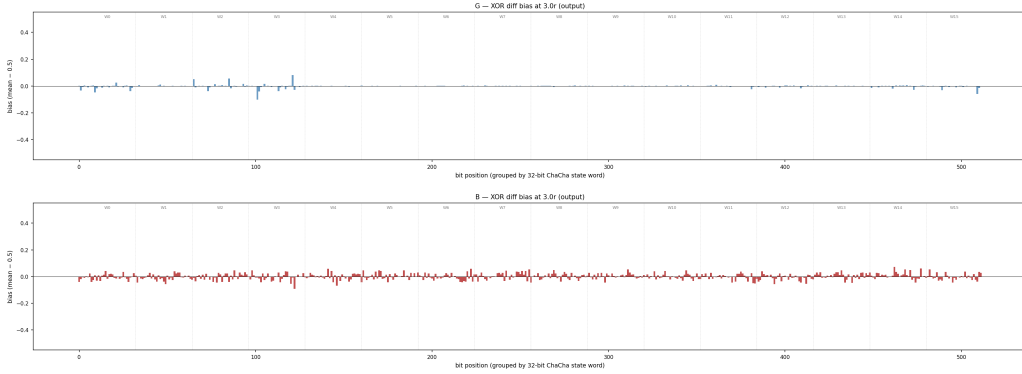


Figure 2: Per-bit output XOR-difference bias at 3 rounds, over the high-confidence set G (top) and the low-confidence set B (bottom). Both are near-flat.

The signal lives in the early-round state Partially decrypting the ciphertexts of G to recover the intermediate states reached *after* one and two rounds, and recomputing the per-bit XOR-difference bias there, tells a very different story (Figure 3). After the first round, 498 of 512 bits have $|\text{bias}| \approx 0.5$ essentially deterministic, and after the second round 262 bits remain strongly constrained ($|\text{bias}| > 0.3$). The strong differential structure created in the first round is therefore still largely intact after two rounds; it is only the third (final) round that washes it out at the bit level.

Truncated differentials

The intermediate-state analysis reveals that the distinguishing information is concentrated in the early rounds. Thresholding the intermediate bias at $|\text{bias}| > 0.3$ turns the constrained bits into truncated-differential (TD) masks (Figure 4): TD_{1r} fixes the 498 bits constrained after one round and is satisfied by 69.3% of G , while the tighter TD_{2r} fixes the 262 bits constrained after two rounds and is satisfied by 7.6% of G . We verify these masks on

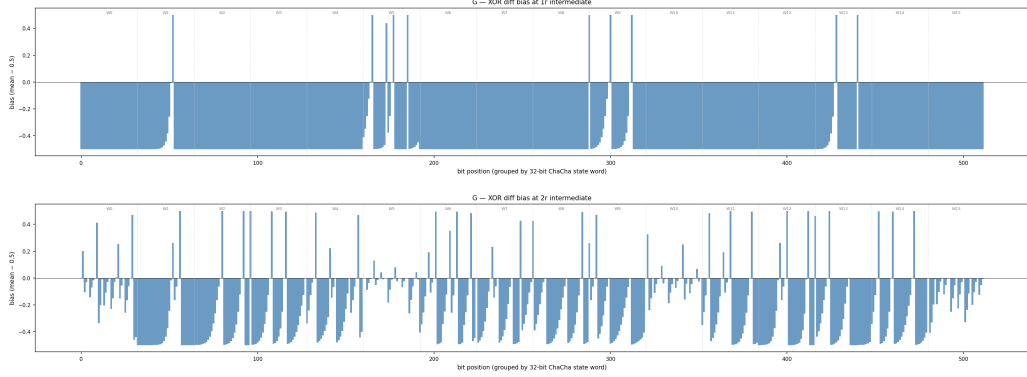


Figure 3: Per-bit XOR-difference bias over G at the intermediate state after one round (top) and after two rounds (bottom). The bias is near-deterministic after the first round.

fresh data: one million previously unseen real pairs, filtered by mask and rescored: pairs satisfying TD_{2r} are classified real with accuracy 1.0000, and those satisfying the broader TD_{1r} with accuracy 0.9928. The first is effectively a *sufficient* condition for the network’s “real” decision; the second nearly matches the network on its own.

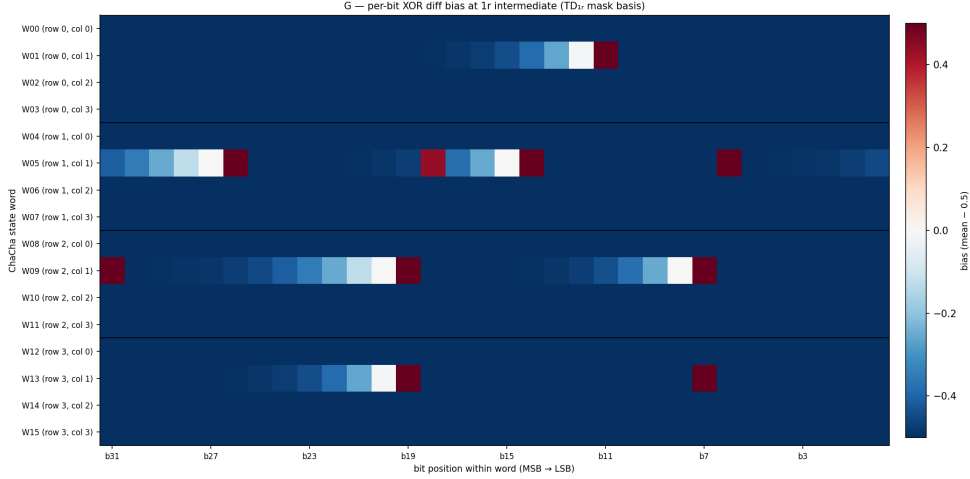


Figure 4: The TD_{1r} truncated-differential mask as 16×32 constrained bits.

A mixture of truncated differentials

A single truncated differential does not explain the complete set of high-confidence pairs. We therefore apply an iterative discovery procedure. Starting with the high-confidence set G , we derive a truncated-differential condition, remove the pairs satisfying that condition, and repeat the procedure on the residual set. At the state after the first round, *five* disjoint truncated differentials cover 99.0% of G , with successive coverages 69.3%, 16.6%, 9.1%, 2.6%, 1.5% (Figure 5).

A threshold sweep (Figure 6) confirms the robustness of the picture: across bias thresholds from 0.10 to 0.40, TD_{2r} -filtered pairs are always classified real with accuracy ≥ 0.9998 , while TD_{1r} tracks the baseline network accuracy of ≈ 0.988 .

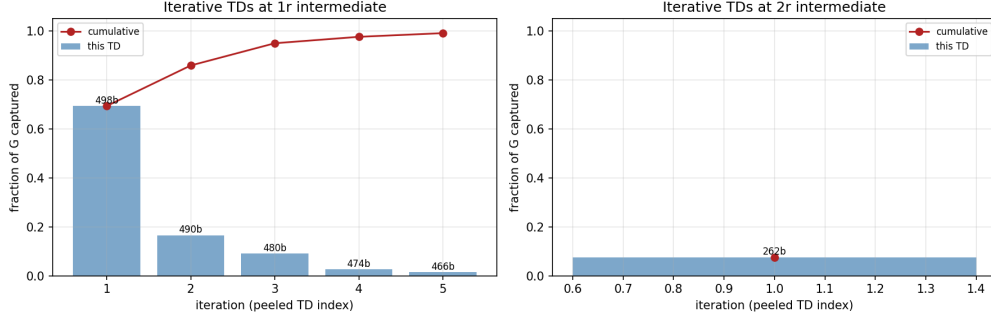


Figure 5: Iterative truncated-differential discovery on G . Five disjoint one-round TDs cover 99.0% of the high-confidence pairs.

Interpretation

The four experiments compose into a coherent account. First, the real-differences experiment demonstrates that the classifier is not exclusively dependent on the output XOR difference. Second, output-difference-only training shows that the output difference nevertheless contains most of the distinguishing information. Third, bit sampling demonstrates that only a minority of the 512 state bits are necessary to recover most of the classifier’s performance.

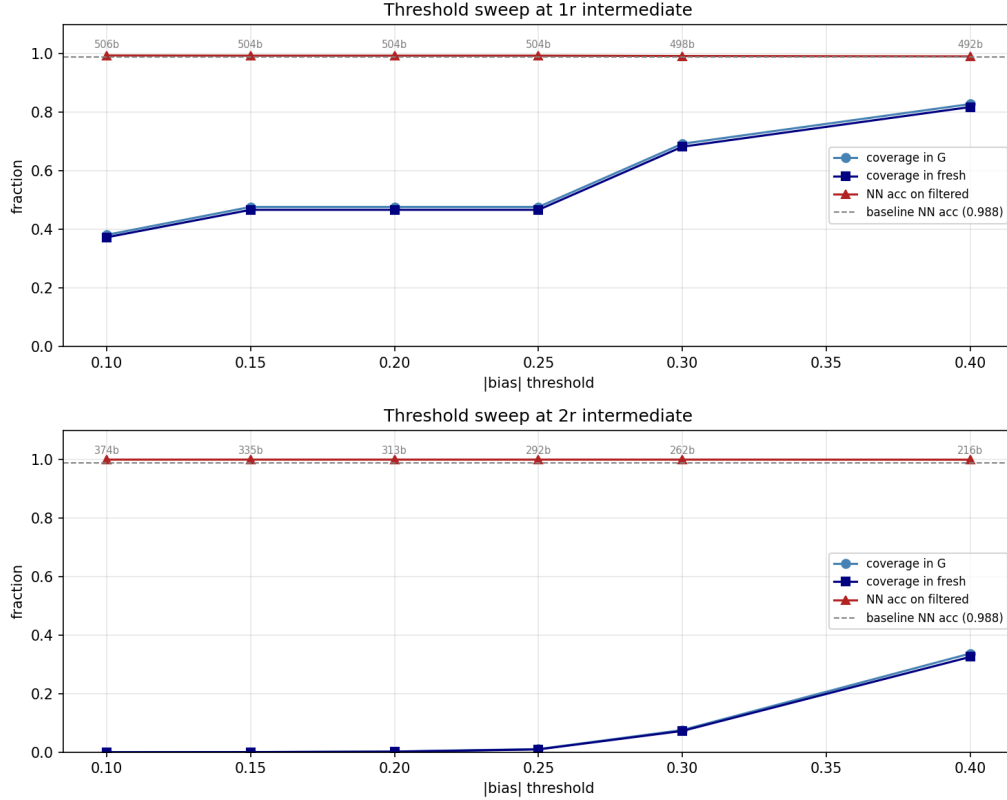


Figure 6: Bias-threshold sweep. TD_{2r} is a near-perfect sufficient condition at every threshold; TD_{1r} tracks the baseline network accuracy.

Finally, intermediate-state analysis reveals that these informative features are strongly associated with differential structure created during the first and second rounds.

The network does not read the ciphertext output difference directly, since that difference is diffused flat at three rounds. Instead, its decision is governed by a small union of truncated differentials at the state reached after the first round, of which the tighter after-two-rounds TD (7.6% coverage, 100% verified accuracy) is one prominent sub-pattern. The 3-round ChaCha distinguisher is thus interpretable as a **learned mixture of round-1 truncated differentials**, refined by absolute-value context consistent with the real-differences and output-difference-only results above.

This interpretation should be regarded as an empirical characterization of the learned classifier rather than as a proof that the neural network implements an explicit logical combination of these truncated differentials.

8 Future Works

In this work, we have mainly focused on distinguishing attacks, a major aspect of the analysis of Differential-neural cryptanalysis, but there are still several directions that remain for future research. The primary goal is to extend this proposed neural distinguishing attack toward *key-recovery attacks*, where the learned differential and truncated-differential structures are used to rank or filter key candidates. Further work can also explore automated input-difference search, additional neural architectures, and joint multi-pair processing. Finally, extending the explainability analysis to more rounds and to Salsa and Forró may reveal additional cryptographic structures that can be exploited for practical neural-assisted key recovery.

Moreover, to the best of our knowledge, differential-neural cryptanalysis has not yet been explored for LFSR-based stream ciphers. Our work aims to bridge this gap by integrating neural-network-based techniques with classical cryptanalytic approaches, particularly differential and differential-linear methods. This will provide a new perspective for analyzing the security of LFSR-based constructions and may enable the identification of statistical dependencies and weaknesses that are difficult to capture using conventional techniques alone. The proposed research will therefore lay a foundation for developing neural-assisted classical cryptanalysis, opening up new directions for the analysis of lightweight and stream-cipher designs. In the longer term, the framework could be extended to other symmetric-key primitives and hybrid cryptanalytic methodologies.

9 Conclusion

We have presented a systematic differential-neural study of the ARX stream cipher ChaCha. Across six neural architectures, three round counts, and five multi-pair settings, we found that 3-round ChaCha is a sharp wall: structured convolutional and attention models distinguish it with accuracy near 0.99 (rising to 1.0 under the multi-pair rule), the structure-agnostic MLP stays at chance, and all models collapse by 3.5 rounds. Applying the same suite to Salsa and Forró places their walls at ≈ 4.5 and ≈ 1.25 rounds, respectively, quantifying the much faster diffusion of Forró’s five-input quarter-round.

Our controlled round-1-weakened variants, linearizing the first round and changing the rotation constants, shift the distinguishable boundary by a measurable fraction of a round and thereby isolate the contribution of the first round and of the rotation amounts to diffusion. Finally, the explainability analysis shows that the trained 3-round ChaCha distinguisher, while built chiefly on the output difference, is not a pure differential distinguisher: its decision is interpretable as a learned mixture of round-1 truncated differentials, five of which already cover 99% of its high-confidence pairs. These results

extend differential-neural cryptanalysis from its block-cipher heartland to the ARX stream-cipher setting and give a concrete cryptographic reading of what such a distinguisher learns.

Overall, our results suggest that neural differential cryptanalysis should not be viewed solely as a black-box replacement for classical statistical distinguishers. Neural networks can instead provide a mechanism for discovering and combining structured cryptanalytic signals that may be difficult to capture with a single manually designed differential condition. This observation motivates closer integration between neural methods and classical differential, truncated-differential, and differential-linear cryptanalysis.

References

- [AFK⁺08] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of salsa, chacha, and rumba. In *Fast Software Encryption*, volume 5086 of *Lecture Notes in Computer Science*, pages 470–488. Springer, 2008. doi:10.1007/978-3-540-71039-4_30.
- [Bak22] Anubhab Baksi. Machine learning-assisted differential distinguishers for lightweight ciphers. In *Classical and Physical Security of Symmetric Key Cryptographic Algorithms*, pages 141–162. Springer, 2022. doi:10.1007/978-981-16-6522-6_6.
- [BBP22] Nicoleta-Norica Bacuieti, Lejla Batina, and Stjepan Picek. Deep neural networks aiding cryptanalysis: A case study of the speck distinguisher. In *Applied Cryptography and Network Security*, volume 13269 of *Lecture Notes in Computer Science*, pages 809–829. Springer, 2022.
- [Ber08a] Daniel J. Bernstein. Chacha, a variant of salsa20. In *Workshop Record of SASC 2008*, pages 3–5, 2008.
- [Ber08b] Daniel J. Bernstein. The salsa20 family of stream ciphers. In Matthew Robshaw and Olivier Billet, editors, *New Stream Cipher Designs: The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 84–97. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-68351-3_8.
- [BGHR23] Emanuele Bellini, David G rault, Anna Hambitzer, and Matteo Rossi. A cipher-agnostic neural training pipeline with automated finding of good input differences. *IACR Transactions on Symmetric Cryptology*, 2023(3):184–212, 2023. doi:10.46586/tosc.v2023.i3.184-212.
- [BGL⁺22] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Enhancing differential-neural cryptanalysis. In *Advances in Cryptology – ASIACRYPT 2022*, Lecture Notes in Computer Science. Springer, 2022.
- [BGPT21] Adrien Benamira, David G rault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In Anne Canteaut and Fran ois-Xavier Standaert, editors, *Advances in Cryptology – EUROCRYPT 2021, Part I*, volume 12696 of *Lecture Notes in Computer Science*, pages 805–835, Cham, 2021. Springer. doi:10.1007/978-3-030-77870-5_28.
- [BLT20] Christof Beierle, Gregor Leander, and Yosuke Todo. Improved differential-linear attacks with applications to arx ciphers. In *Advances in Cryptology – CRYPTO 2020*, volume 12172 of *Lecture Notes in Computer Science*. Springer, 2020.

- [BLYZ23] Zhenzhen Bao, Jinyu Lu, Yiran Yao, and Liu Zhang. More insight on deep learning-aided cryptanalysis. pages 436–467, 2023. doi:[10.1007/978-981-99-8727-6_15](https://doi.org/10.1007/978-981-99-8727-6_15).
- [BR20] Emanuele Bellini and Matteo Rossi. Performance comparison between deep learning-based and conventional cryptographic distinguishers. *IACR Cryptology ePrint Archive*, 2020. arXiv:[2020/953](https://arxiv.org/abs/2020.0953).
- [BS91] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. *Journal of Cryptology*, 4(1):3–72, 1991. doi:[10.1007/BF00630563](https://doi.org/10.1007/BF00630563).
- [CM17] Arka Rai Choudhuri and Subhamoy Maitra. Significantly improved multi-bit differentials for reduced round salsa and chacha. *IACR Transactions on Symmetric Cryptology*, 2017.
- [CPdSJB20] Murilo Coutinho, Iago Passos, Rafael de Sousa Junior, and Fábio Borges. Improving the security of chacha against differential-linear cryptanalysis. pages 15–28, 10 2020. doi:[10.5753/sbseg.2020.19224](https://doi.org/10.5753/sbseg.2020.19224).
- [CPV⁺23] Matheus Coutinho, Igor Passos, José Carlos Grados Vázquez, Felipe Luiz Lima de Mendonça, Rafael Timóteo de Sousa Jr., and Fernando Borges. Latin dances reloaded: Improved cryptanalysis against salsa and chacha, and the proposal of forró. In *Advances in Cryptology – ASIACRYPT 2022*, volume 13791 of *Lecture Notes in Computer Science*, pages 256–286, Cham, 2023. Springer. doi:[10.1007/978-3-031-22972-5_9](https://doi.org/10.1007/978-3-031-22972-5_9).
- [CSY23] Yuan Chen, Yifan Shen, and Hong Yu. Neural-aided statistical attack for cryptanalysis. *The Computer Journal*, 66(10):2480–2498, 2023.
- [CSYY23] Yi Chen, Yantian Shen, Hongbo Yu, and Sitong Yuan. A new neural distinguisher considering features derived from multiple ciphertext pairs. *The Computer Journal*, 66(6):1419–1433, 2023. doi:[10.1093/comjnl/bxac019](https://doi.org/10.1093/comjnl/bxac019).
- [DCC23] Haoran Deng, Xianghui Cao, and Yu Cheng. Attention in differential cryptanalysis on lightweight block cipher speck. In *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, pages 1–9, 2023. doi:[10.1109/PST58708.2023.10320201](https://doi.org/10.1109/PST58708.2023.10320201).
- [DGSS22] Sabyasachi Dey, Hirendra Kumar Garai, Santanu Sarkar, and Nitin Kumar Sharma. Revamped differential-linear cryptanalysis on reduced round chacha. *IACR Cryptol. ePrint Arch.*, 2022:536, 2022. URL: <https://eprint.iacr.org/2022/536>.
- [DLS25] Sabyasachi Dey, Gregor Leander, and Nitin Kumar Sharma. Improved key recovery attacks on reduced-round salsa20. *Des. Codes Cryptogr.*, 93(1):243–262, 2025. URL: <https://doi.org/10.1007/s10623-024-01522-7>, doi:[10.1007/S10623-024-01522-7](https://doi.org/10.1007/S10623-024-01522-7).
- [DMSS25] Sabyasachi Dey, Subhamoy Maitra, Santanu Sarkar, and Nitin Kumar Sharma. Significantly improved cryptanalysis of salsa20 with two-round criteria. *IACR Trans. Symmetric Cryptol.*, 2025(1):420–443, 2025. URL: <https://doi.org/10.46586/tosc.v2025.i1.420-443>, doi:[10.46586/TOSC.V2025.I1.420-443](https://doi.org/10.46586/TOSC.V2025.I1.420-443).
- [GHHP26] David Gerault, Anna Hambitzer, Moritz Huppert, and Stjepan Picek. Survey: Six years of neural differential cryptanalysis. *IACR Communications in Cryptology*, 3(2), 2026. doi:[10.62056/a69qxruc2](https://doi.org/10.62056/a69qxruc2).

- [GLN22] Aron Gohr, Gregor Leander, and Patrick Neumann. An assessment of differential-neural distinguishers. Cryptology ePrint Archive, Report 2022/1521, 2022. URL: <https://eprint.iacr.org/2022/1521>.
- [GLP⁺24] Yue-Tian Goi, Shu-Min Leong, Raphaël C.-W. Phan, Shangqi Lai, and Ana Sălăgean. Unveiling the black box: Neural cryptanalysis with xai. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1951–1956, 2024. doi:10.1109/SMC54092.2024.10831564.
- [Goh19] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, volume 11693 of *Lecture Notes in Computer Science*, pages 150–179, Cham, 2019. Springer. doi:10.1007/978-3-030-26951-7_6.
- [HGH⁺23] Anna Hambitzer, David Gérault, Yun Ju Huang, Najwa Aaraj, and Emanuele Bellini. Nnbits: Bit profiling with a deep learning ensemble based distinguisher. In *Topics in Cryptology – CT-RSA 2023*, volume 13819 of *Lecture Notes in Computer Science*, pages 493–523. Springer, 2023. doi:10.1007/978-3-031-30872-7_19.
- [HLF⁺24] T. Huang, Y. Li, Q. Fu, Y. Chen, and L. Song. Improving differential-neural cryptanalysis for large-state speck. In *Information and Communications Security – 26th International Conference, ICICS 2024*, volume 15056 of *Lecture Notes in Computer Science*, pages 40–57. Springer, 2024. doi:10.1007/978-981-97-8798-2_3.
- [LJSC24] Guangqiu Lv, Chenhui Jin, Zhen Shi, and Ting Cui. Approximating neural distinguishers using differential-linear imbalance. *The Journal of Supercomputing*, 80(19):26865–26889, 2024. doi:10.1007/s11227-024-06375-4.
- [LRC23] J. S. Liu, J. J. Ren, and S. Z. Chen. A deep learning aided differential distinguisher improvement framework with more lightweight and universality. *Cybersecurity*, 6(1), 2023. doi:10.1186/s42400-023-00176-7.
- [MPKM⁺22] Girish Mishra, S. K. Pal, S. V. S. S. N. V. G. Krishna Murthy, Ishan Prakash, and Anshul Kumar. Deep learning-based differential distinguisher for lightweight ciphers gift-64 and pride. In *Machine Intelligence and Smart Systems*, pages 245–257. Springer, 2022. doi:10.1007/978-981-16-9650-3_19.
- [PMC⁺22] Debranjana Pal, Upasana Mandal, Mainak Chaudhury, Abhijit Das, and Dipanwita Roy Chowdhury. A deep neural differential distinguisher for ARX based block cipher. Cryptology ePrint Archive, Paper 2022/1195, 2022. URL: <https://eprint.iacr.org/2022/1195>.
- [Seo24] Byoungjin Seok. Truncated differential-neural key recovery attacks on round-reduced hight. *Electronics*, 13(20):4053, 2024. doi:10.3390/electronics13204053.
- [WWH21] Gao Wang, Gaoli Wang, and Yihui He. Improved machine learning assisted (related-key) differential distinguishers for lightweight ciphers. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 164–171. IEEE, 2021. doi:10.1109/TrustCom53373.2021.00037.

- [ZWw22] Liu Zhang, Zilong Wang, and Baocang wang. Improving differential-neural cryptanalysis. Cryptology ePrint Archive, Paper 2022/183, 2022. URL: <https://eprint.iacr.org/2022/183>.