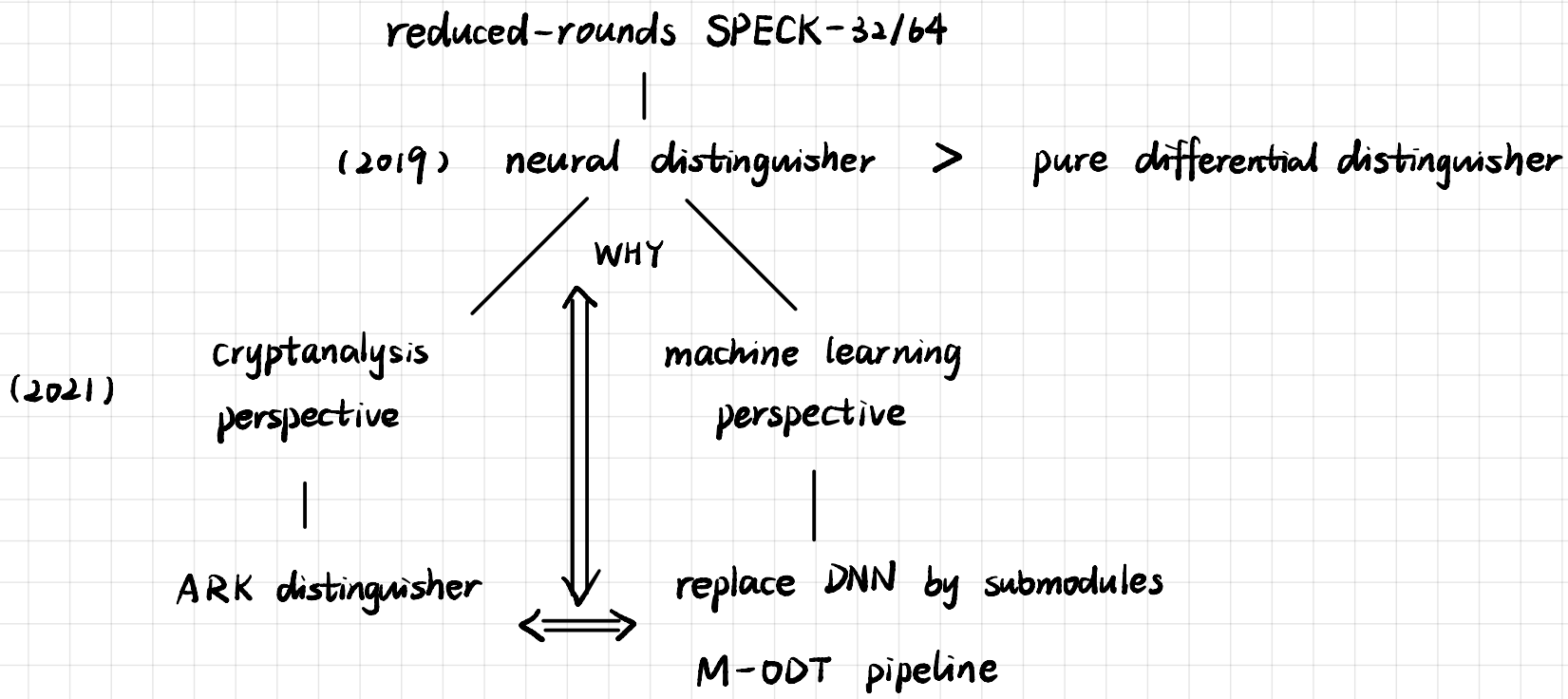
# A Deeper Look at Machine Learning-Based Cryptanalysis

Yichen Zhu
05/29/2021

# What is this article doing?



# Preliminaries

## Basic Notations

$\oplus$ : XOR operation

$\wedge$ : AND operation

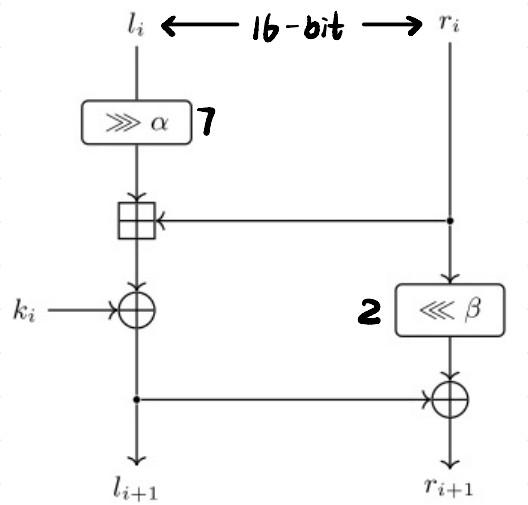
$\boxplus$ : modular addition

$\ggg$ : left bit rotation

$\lll$ : right bit rotation

$a \| b$ : concatenation of two bit strings $a$ and $b$

## SPECK-32/64




$l_0 \| r_0 \leftarrow P$

$C \leftarrow l_{22} \| r_{22}$

$V_i = l_i \oplus r_i$

$\Delta L = C_l \oplus C'_l$

$\Delta R = C_r \oplus C'_r$

## Differential Cryptanalysis

$f : \mathbb{F}_2^b \rightarrow \mathbb{F}_2^b$ given $\Delta x, \Delta y$

$$P(\Delta x \xrightarrow{f} \Delta y) = \frac{\#\{x \mid f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^b}$$

classical tool : DDT (Difference Distribution Table)

$\forall (\Delta x, \Delta y) \in \mathbb{F}_2^b \times \mathbb{F}_2^b$ : $DDT(\Delta x, \Delta y) = P(\Delta x \xrightarrow{f} \Delta y)$

## Deep Neural Networks (DNN)

given dataset $D = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$

$$\text{find } \theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{n} \text{Loss}(DNN_\theta(x_i), y_i)$$

DNN is powerful to derive non-linear features from training data.

easily influented by noise

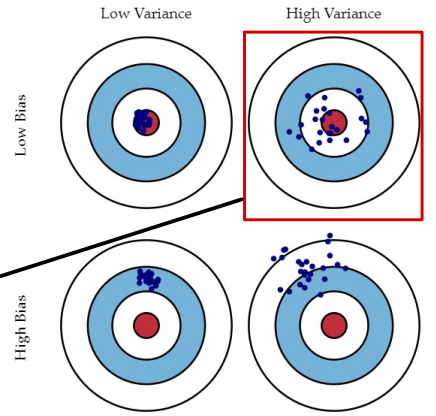



Fig. 1 Graphical illustration of bias and variance.

# A First Look at Gohr's CRYPTO 2019 Results

10.1007/978-3-030-26951-7_6

**Overview** Gohr tried to distinguish $\begin{cases} \text{real pairs} & \{(C,C') \mid \Delta P = 0x0040/0000\} \\ \text{random pairs} & \{(C,C')\} \end{cases}$

## Pure Differential Distinguishers

1. Gohr computes $DDT(0x0040/0000, *)$ using the **Markov assumption**.

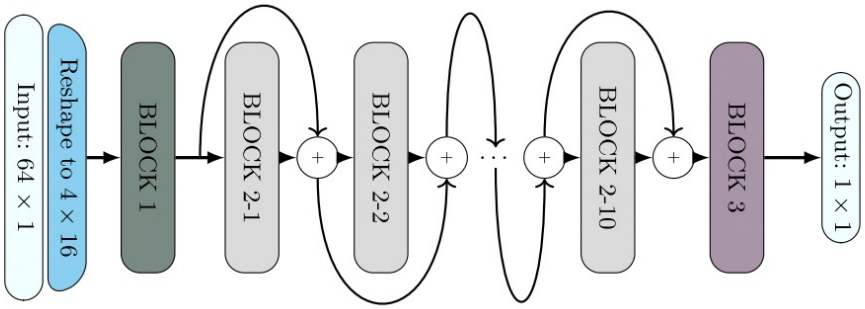
2. $D_{nr}(C,C') := \begin{cases} (C,C') \text{ is a real pair}, & DDT(0x0040/0000 \to \Delta C) > \frac{1}{2^{32}-1} \\ & \quad \| \\ & \quad C \oplus C' \\ (C,C') \text{ is a random pair}, & \text{otherwise} \end{cases}$

number of rounds

## Gohr's Neural Distinguisher

$(C, C')$

$(C_l, C_r, C'_l, C'_r)$




$0 \le \text{score} \le 1$

$N_{nr}(C,C') := \begin{cases} (C,C') \text{ is a real pair}, & \text{score} \ge 0.5 \\ (C,C') \text{ is a random pair}, & \text{otherwise} \end{cases}$

## Experiment Results

| Rds | Distinguisher | Accuracy | TPR | TNR |
|---|---|---|---|---|
| 5 | $D_5$ | 0.911 | 0.877 | 0.947 |
| | $N_5$ | $0.929 \pm 5.13 \times 10^{-4}$ | $0.904 \pm 8.33 \times 10^{-4}$ | $0.954 \pm 5.91 \times 10^{-4}$ |
| 6 | $D_6$ | 0.758 | 0.680 | 0.837 |
| | $N_6$ | $0.788 \pm 8.17 \times 10^{-4}$ | $0.724 \pm 1.26 \times 10^{-3}$ | $0.853 \pm 1.00 \times 10^{-3}$ |
| 7 | $D_7$ | 0.591 | 0.543 | 0.640 |
| | $N_7$ | $0.616 \pm 9.7 \times 10^{-4}$ | $0.533 \pm 1.41 \times 10^{-3}$ | $0.699 \pm 1.30 \times 10^{-3}$ |
| 8 | $D_8$ | 0.512 | 0.496 | 0.527 |
| | $N_8$ | $0.514 \pm 1.00 \times 10^{-3}$ | $0.519 \pm 1.41 \times 10^{-3}$ | $0.508 \pm 1.42 \times 10^{-3}$ |

TPR: True Positive Rate

TNR: True Negative Rate

⇒ $N_{nr}$ outperforms $D_{nr}$, $N_{nr}$ may learn something more than $D_{nr}$

## Real Difference Experiment

challenge: distinguish ⟨ real pairs $\{(C, C') \mid \Delta P = 0x0040/0000\}$

masked real pairs $\{(C \oplus M, C' \oplus M) \mid \Delta P = 0x0040/0000, \text{ random } M\}$

⇓

why do this exepriment ?

$(C \oplus M) \oplus (C' \oplus M) = C \oplus C'$ ⇒ masked real pairs do not affect the output difference distribution

without retraining $N_{nr}$ :

| Rds | Distinguisher | Accuracy |
|---|---|---|
| 5 | $N_5$ | $0.707 \pm 9.10 \times 10^{-4}$ |
| 6 | $N_6$ | $0.606 \pm 9.77 \times 10^{-4}$ |
| 7 | $N_7$ | $0.551 \pm 9.95 \times 10^{-4}$ |
| 8 | $N_8$ | $0.507 \pm 1.00 \times 10^{-3}$ |

⇒ $N_{nr}$ is still able to distinguish

⇓

$N_{nr}$ do not just rely on the difference distribution

# Interpretation of Gohr's Neural Network: a Cryptanalysis Perspective

**Question** What type of cryptanalysis is Gohr's neural distinguisher learning ?

⇓

find common patterns in real pairs

## Choice of Input Difference

$0x0040/0000$

1. part of a 9-round differential characteristics [1]
2. Gohr : $0x0040/0000$ transits with low Hamming weight

for 5 rounds : $\max P(0x0040/0000 \to *) = P(0x0040/0000 \to 0x802a/d4a8) \approx 2^{-13}$

$\max P(* \to *) = P(0x2800/0010 \to 0x850a/9520) \approx 2^{-9}$

distinguish { real pairs $\{(C, C') \mid \Delta P = 0x2800/0010\}$ ; random pairs $\{(C, C')\}$ } ⇒ $N_{nr}$ performs badly ! (retrained)

## Question

How to find "$0x0040/0000$" in other ciphers ?

Is there an algorithm ?

global optimization ? local optimization ?

[1] Abed, F., List, E., Lucks, S., Wenzel, J.: Differential cryptanalysis of round-reduced simon and speck. In: Fast Software Encryption - FSE 2014. LNCS, vol. 8540, pp. 525–545. Springer (2014)

# Changing the Inputs to the Neural Network

distinguish { real pairs $\{(C, C') \mid \Delta P = 0x0040/0000\}$ ; random pairs $\{(C, C')\}$

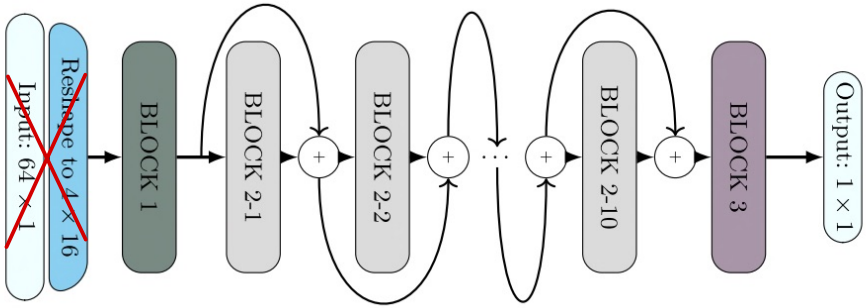
$$D_{nr}(C, C') := \begin{cases} (C, C') \text{ is a real pair}, & DDT(0x0040/0000 \to \underbrace{\Delta C}_{= C \oplus C'}) > \frac{1}{2^{32}-1} \\ (C, C') \text{ is a random pair}, & \text{otherwise} \end{cases}$$

~~$(C, C')$~~

$\Downarrow$

$C \oplus C'$

retrain




| Rds | Distinguisher | Accuracy | TPR | TNR |
|---|---|---|---|---|
| 5 | $D_5$ | 0.911 | 0.877 | 0.947 |
| | $N_5$ | $0.929 \pm 5.13 \times 10^{-4}$ | $0.904 \pm 8.33 \times 10^{-4}$ | $0.954 \pm 5.91 \times 10^{-4}$ |
| 6 | $D_6$ | 0.758 | 0.680 | 0.837 |
| | $N_6$ | $0.788 \pm 8.17 \times 10^{-4}$ | $0.724 \pm 1.26 \times 10^{-3}$ | $0.853 \pm 1.00 \times 10^{-3}$ |
| 7 | $D_7$ | 0.591 | 0.543 | 0.640 |
| | $N_7$ | $0.616 \pm 9.7 \times 10^{-4}$ | $0.533 \pm 1.41 \times 10^{-3}$ | $0.699 \pm 1.30 \times 10^{-3}$ |
| 8 | $D_8$ | 0.512 | 0.496 | 0.527 |
| | $N_8$ | $0.514 \pm 1.00 \times 10^{-3}$ | $0.519 \pm 1.41 \times 10^{-3}$ | $0.508 \pm 1.42 \times 10^{-3}$ |

Changed ACC

0.906

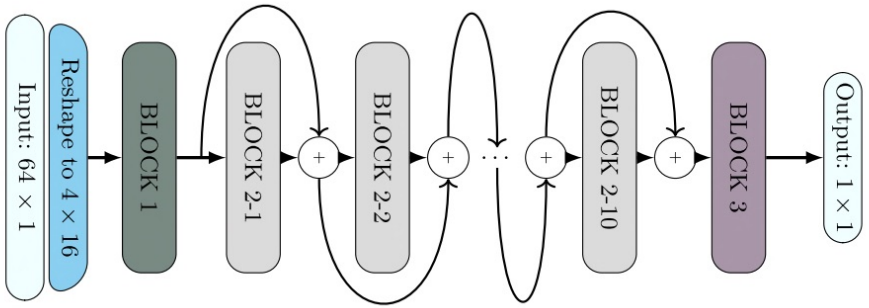
0.754

0.583

$\Rightarrow$ similar to D

$\Rightarrow$ $N_{nr}$ learns more than the distribution of the differences

# Analyzing Ciphertext Pairs




$0 \le score \le 1$

$G, B \subseteq \{real\ pairs\}$ Good, Bad

$G = \{(C, C') \mid score \ge 0.9\}$

$B = \{(C, C') \mid score \le 0.1\}$

## Experiment A ⇒ focus on differentials

(1) $10^5$ $\{(C, C') \mid \Delta P = 0x0040/0000\}$

↓

(2) $G = \{(C, C') \mid score \ge 0.9\}$

↓

(3) $Diff_5 = \{\Delta C^{(1)}, \Delta C^{(2)}, \cdots, \Delta C^{(m)}\}$ sort by frequency

(4) $\forall \Delta C^{(i)} \in Diff_5$

generate $10^4$ $\{(P, P') \mid \Delta P = \Delta C^{(i)}\}$

↓ $C = P, C' = P'$

(5) $\{(C, C') \mid score \ge 0.5\}$

| No. | Difference | Cnt | Percent. | No. | Difference | Cnt | Percent. |
|---|---|---|---|---|---|---|---|
| 1 | 0x802a/d4a8 | 116 | 75 | 14 | 0x883a/dcb8 | 45 | 75 |
| 2 | 0x802e/d4ac | 81 | 76 | 15 | 0x801e/d49c | 45 | 75 |
| 3 | 0x803a/d4b8 | 73 | 74 | 16 | 0xa026/f4a4 | 42 | 75 |
| 4 | 0x8e2a/daa8 | 73 | 75 | 17 | 0xbe1a/ea98 | 41 | 75 |
| 5 | 0x822a/d6a8 | 72 | 75 | 18 | 0x821a/d698 | 41 | 76 |
| 6 | 0xb82a/eca8 | 67 | 75 | 19 | 0xbe26/eaa4 | 41 | 75 |
| 7 | 0x882a/dca8 | 65 | 75 | 20 | 0x83ea/d768 | 40 | 75 |
| 8 | 0x801a/d498 | 62 | 75 | 21 | 0x8626/caa4 | 40 | 38 |
| 9 | 0xa02a/f4a8 | 62 | 75 | 22 | 0x886a/dce8 | 40 | 75 |
| 10 | 0xbe2a/eaa8 | 62 | 75 | 23 | 0xa06a/f4e8 | 40 | 75 |
| 11 | 0x806a/d4e8 | 59 | 74 | 24 | 0x8e1a/da98 | 39 | 75 |
| 12 | 0x8e26/daa4 | 47 | 75 | 25 | 0x8226/cea4 | 38 | 37 |
| 13 | 0x8026/d4a4 | 46 | 74 | | | | |

⇒ N does not recognize by more probable $\Delta C$

Experiment B ⇒ focus on differences at round 3 and round 4

(1) $10^5$ $\{(C, C') \mid \Delta P = 0x0040/0000\}$

↓

(2) $G = \{(C, C') \mid \text{score} \geq 0.9\}$

↓ decrypt i rounds

(3) $Diff_{5-i} = \{\Delta C_{5-i}^{(1)}, \Delta C_{5-i}^{(2)}, \cdots, \Delta C_{5-i}^{(n)}\}$ sort by frequency

(4) generate $10^5$ $\{(C_4, C_4') \mid \Delta P = 0x0040/0000\}$

(all 4 → 5-i)

↓

(5) $\{(C_4, C_4') \mid \Delta C_4 \in Diff_{5-i}\}$

↓

(6) $\{(C, C') \mid \text{score} \geq 0.5\}$

| | | score ≥ 0.5 | TPR |
|---|---|---|---|
| $i=2$, | $\lvert Diff_{5-i} \rvert = 1669$, $\lvert\{(C_4, C_4') \mid \Delta C_4 \in Diff_{5-i}\}\rvert = 89,969$ | 97.86% | 88.04% $\left(= \frac{89,969 \times 97.86\%}{10^5}\right)$ |
| $i=1$, | $\lvert Diff_{5-i} \rvert = 128,039$, $\lvert\{(C_4, C_4') \mid \Delta C_4 \in Diff_{5-i}\}\rvert = 74,077$ | 99.98% | 74.06% |
| | | $N_5$ | 90.04% |

⇓

not enough

Experiment C ⇒ focus on the bias of the difference bits

bias of $Diff_{5-2}^{G}$ , $Diff_{5-2}^{B}$

| bit position | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $G$ | 0.476 | -0.454 | -0.355 | -0.135 | 0.045 | 0.084 | -0.009 | 0.487 | -0.473 | -0.426 | -0.300 | -0.050 | 0.006 | 0.019 | 0.500 | -0.500 |
| $B$ | -0.002 | 0.018 | 0.008 | -0.011 | 0.044 | 0.002 | 0.023 | -0.022 | 0.010 | -0.002 | 0.013 | -0.004 | 0.006 | -0.005 | 0.103 | 0.072 |

| bit position | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $G$ | 0.476 | -0.454 | -0.142 | -0.006 | 0.025 | 0.084 | -0.009 | 0.487 | -0.473 | -0.426 | 0.165 | 0.094 | -0.006 | 0.019 | -0.500 | -0.500 |
| $B$ | 0.031 | -0.009 | -0.015 | -0.007 | -0.014 | -0.024 | 0.025 | 0.026 | 0.034 | -0.005 | -0.018 | -0.021 | 0.006 | 0.009 | 0.079 | -0.065 |

Truncated Differential

$TD_3$ 3 rounds: $10*****00*****00\ 10*****00*****10$

$TD_4$ 4 rounds: $10*****10*****10\ 10*****10*****00$

(1) generate $10^{6}$ (5) $\{(C_{5-i}, C'_{5-i}) \mid \Delta P = 0x0040/0000\}$

↓

(2) $\{(C, C') \mid \Delta C_{5-i}$ satisfies $TD_i\}$

↓

(3) $\{(C, C') \mid score \geq 0.5\}$

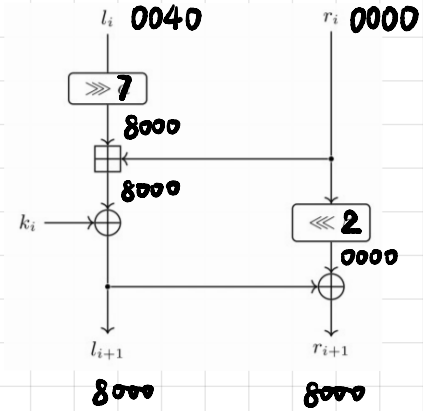
| 5-i | Trunc. Diff. | Dataset size | Acc. | Proport. (TPR) |
|---|---|---|---|---|
| 3 | TD3 | 87741 | 99.277% | 87.11% |
| 4 | TD4 | 50063 | 99.996% | 50.06% |

important point : identify exactly the pairs verified by the neural distinguisher

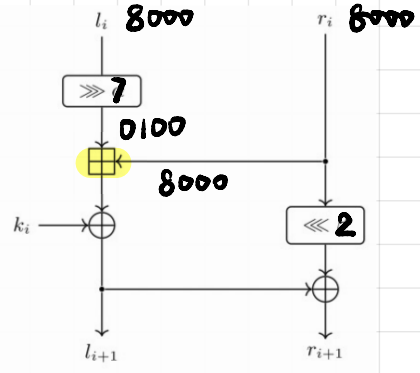
# Deriving $TD_2$ and $TD_4$

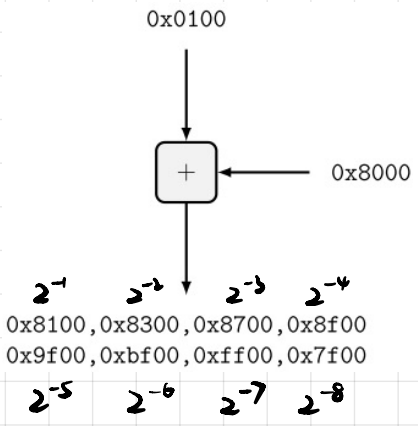
difference propagation 10.1007/978-3-662-04722-4_8

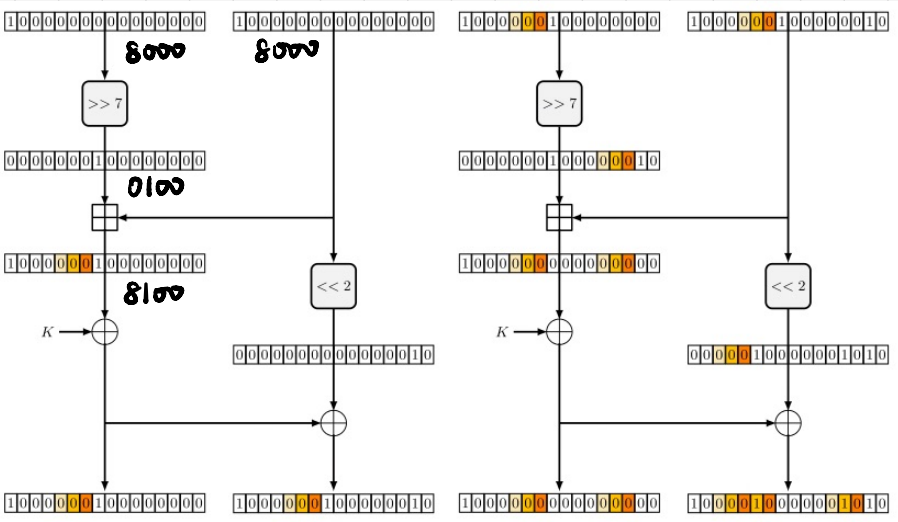
round 1




round 2

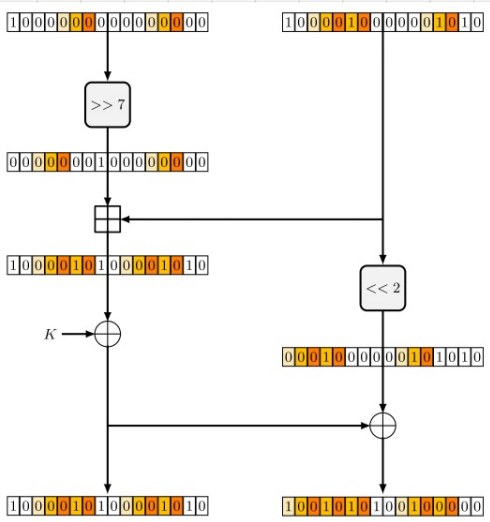









0x8000/8000 → 0x8100/8102  0x8100/8102 → 0x8000/840a  0x8000/840a → 0x850a/9520

$\max P(* \to *)$
$= P(0x2800/0010 \to 0x850a/9520)$
$\approx 2^{-9}$

What does 0x850a/9520 mean?

($TD_4$)

the darker the color, the higher the probability to have a carry to propagate to

Experiment D ⇒ Given a 5-round SPECK-32/64 (C, C')

$N_5$ is able to determine the difference of certain bits at round 3 and 4 with high accuracy

## Average Key Rank Differential Distinguisher

— AKR is not based on machine learning almost gets same accuracy

aDDT a truncated DDT constructed on samples

Idea mask 0xff8f/ff8f ?

(1) use all possible relevant subkey bits to decrypt last round.

(2) compute average probabilities of partial decryptions to get a score

| $N_5$ | Accuracy | TPR | TNR |
|---|---|---|---|
| 5 | 92.98% | 90.76% | 95.22% |
| 6 | 78.79% | 72.53% | 85.07% |
| 7 | 60.28% | 55.31% | 65.24% |

| | | $N_r$ output | |
|---|---|---|---|
| | | $\geq 0.5$ | $< 0.5$ |
| AKR Dist | 1 | 46.6% | 1.48% |
| | 0 | 0.953% | 51.0% |

**Algorithm 1** Function used to build the approximated DDT from a given number of random samples and a bitmask.

**function** BUILDAPPROXDDT($nr, np, m$) ▷ where $nr$ - Number of rounds, $np$ - No. of pairs, $m$ - bitmask
  $aDDT \leftarrow ZEROS(2^{32})$
  **for** $i = 1$ to $np$ **do**
    $p \leftarrow rand(0, 2^{32} - 1)$
    $p' \leftarrow p \oplus$ 0x0040/0000
    $key \leftarrow rand(0, 2^{64} - 1)$
    $l_{nr}, r_{nr} \leftarrow ENC_{key}(p, nr)$;
    $l'_{nr}, r'_{nr} \leftarrow ENC_{key}(p', nr)$
    $\delta_l \leftarrow l_{nr} \oplus l'_{nr}$;
    $\delta_v \leftarrow l_{nr} \oplus r_{nr} \oplus l'_{nr} \oplus r'_{nr}$
    $aDDT[(\delta_l || \delta_v) \wedge m] = aDDT[(\delta_l || \delta_v) \wedge m] + \frac{1}{n}$
  **end for**
  **return** $aDDT$
**end function**

Why $\delta_l$, $\delta_v$ ?

$l_{i-1} \leftarrow l_i$

$r_{i-1} = (l_i \oplus r_i) \ggg 2$

np

**Algorithm 2** Pseudocode for the average key rank differential distinguisher.

**function** AKRDD($nr, C, C', m$) ▷ where $nr$ - Number of rounds, $C$, $C'$ - two ciphertexts, $m$ - bitmask
  $aDDT \leftarrow BuildApproxDDT(nr - 1, 10^7, m)$
  $score \leftarrow 0$
  **for** $key = 0$ to $2^{12} - 1$ **do**
    $l_{nr-1}, r_{nr-1} \leftarrow DEC_{key}(C, 1)$; ▷ Decrypt the last round with the subkey $key$
    $l'_{nr-1}, r'_{nr-1} \leftarrow DEC_{key}(C', 1)$
    $\delta_l \leftarrow l_{nr-1} \oplus l'_{nr-1}$;
    $\delta_v \leftarrow l_{nr-1} \oplus r_{nr-1} \oplus l'_{nr-1} \oplus r'_{nr-1}$
    $score+ = aDDT[\delta_l || \delta_v]$
  **end for**
  **return** $(\frac{score}{2^{12}} > 2^{-HW(m)})$ ▷ HW(x) refers to the Hamming weight of x
**end function**

## Discussion

1. N may learn differential-linear cryptanalysis in the case of SPECK.
   > for AES-2-2-4, N relies fully on differential.
2. N may learn linear characteristics while considering dependencies and correlations
3. N may efficiently learn short but strong differential, linear, or differential-linear characteristics for small block ciphers for a small number of rounds.

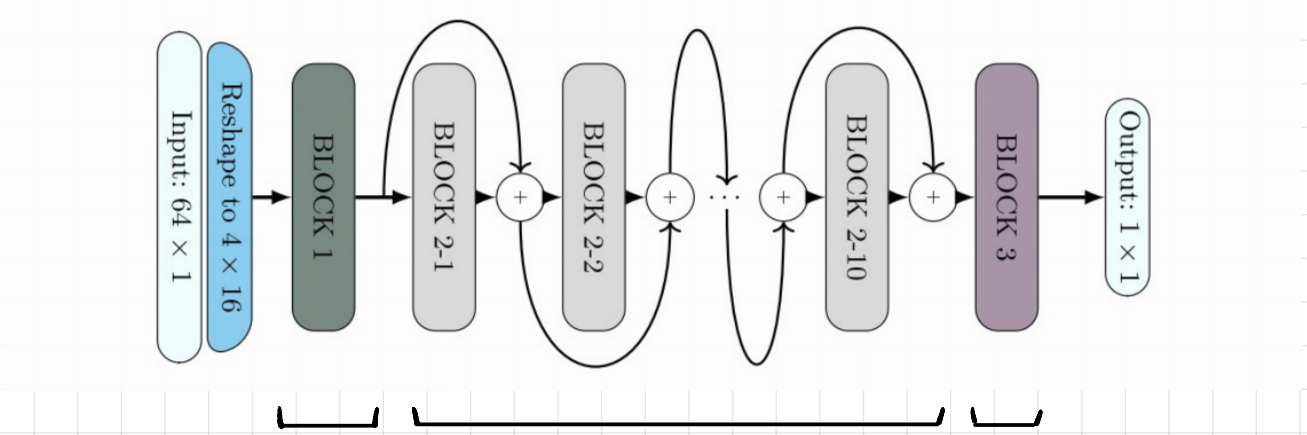
# Interpretation of Gohr's Neural Network: a Machine Learning Perspective

**Question** Can $N$ be replaced by a strategy inspired by ⟨ differential cryptanalysis / machine learning ⟩ ? possible

DNN often outperforms ⟨ mathematical modeling / standard machine learning approaches ⟩ in supervised data-driven settings, especially on high-dimensional data.

$N$ seems to find a property $\mathcal{P}$ currently unknown by cryptanalysts

**Objective** replace submodules of $N$ by interpretable ones



## Replace BLOCK 3

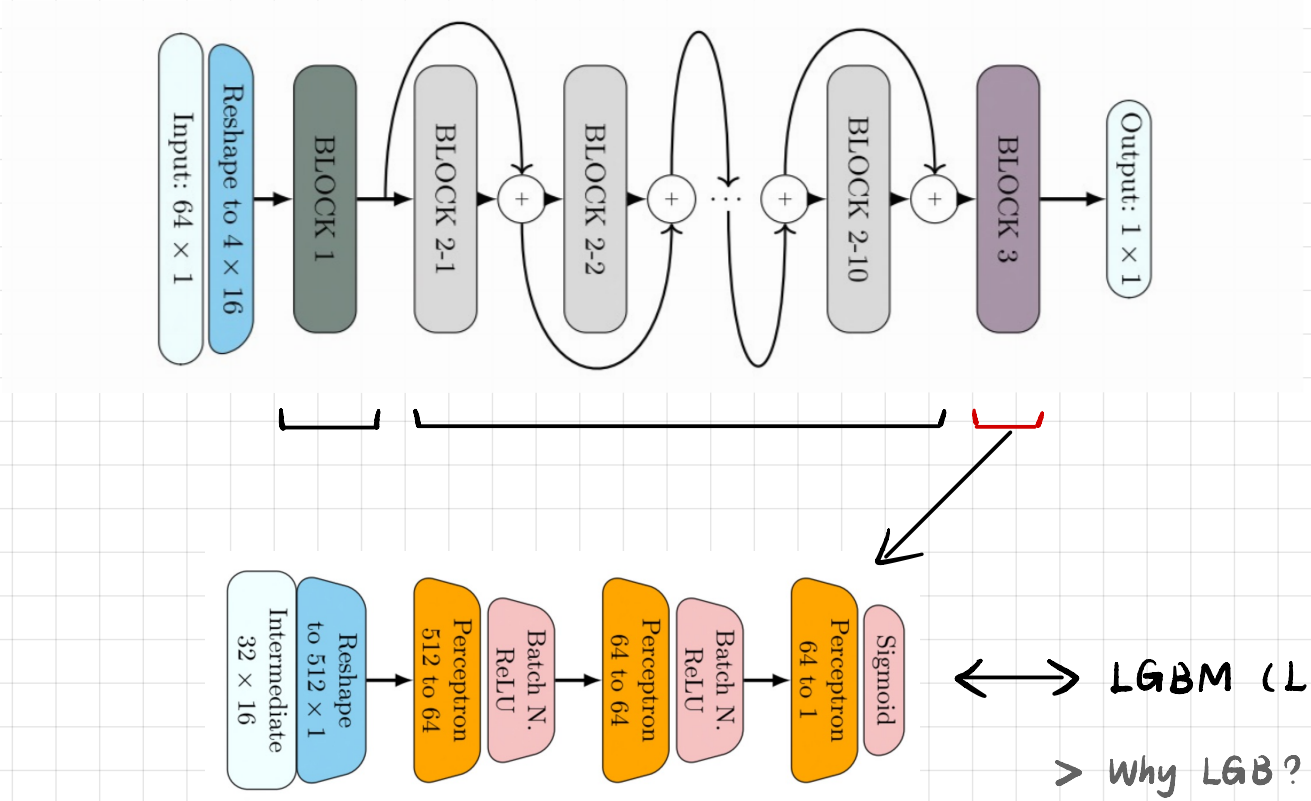



Fig. 5: The classification block (Block 3).

⟷ LGBM (Light Gradient Boosting Machine)

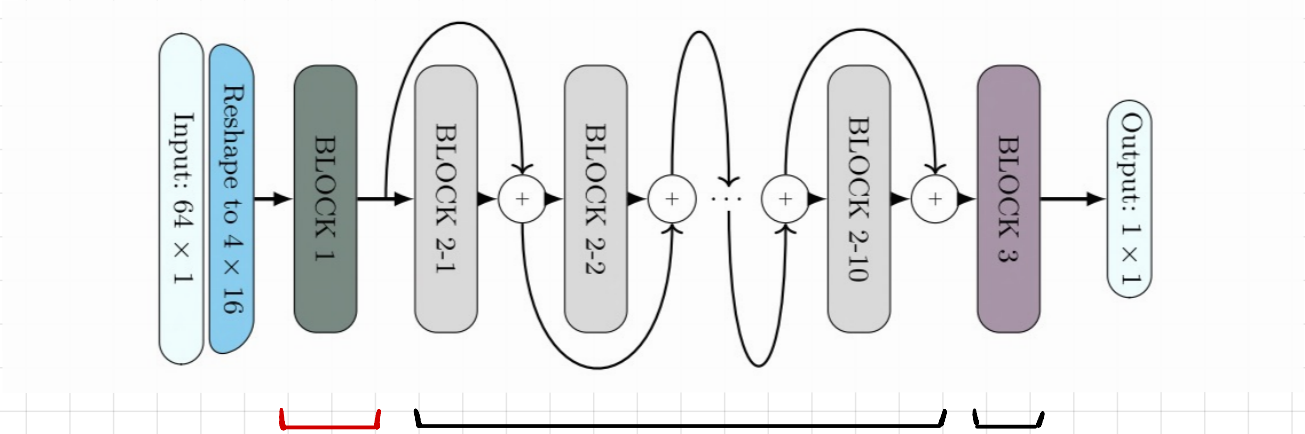
> Why LGB? more accurate, interpretable and faster

| $N_5$ | $D_5$ | LGBM as classifier for the original input | LGBM as classifier for the 512-feature | LGBM as classifier for the 64-feature |
|---|---|---|---|---|
| 92.9% | 91.1% | 76.34% ± 2.62 | 91.49% ± 0.09 | 92.36% ± 0.07 |

⇓ entirely replaced ✗

⇓ partially replaced ✓

## Replace BLOCK 1



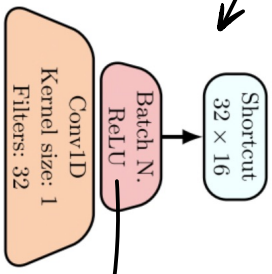




Fig. 3: Initial convolution block (Block 1).

performs a linear transformation on the input

observation : weights contain many opposite values

⇒ DNN is looking for differences between input features

⇒ $(C_e, C_r, C_{\acute{e}}, C_{\acute{r}}) \rightarrow (\Delta L, \Delta V, V_0, V_1)$ and a linear combination of those

input : binary values

output : non-zero value → 1

⇒ truth-table boolean expressions

$$ReLU(x) = \max(0, x)$$

$$H(x) = \frac{1}{2} + \frac{sgn(x)}{2}$$

get almost same ACC and boolean expressions

retrain with $(\Delta L, \Delta V, V_0, V_1)$ as input , get boolean expressions as expected.

## Replace Block 2-i

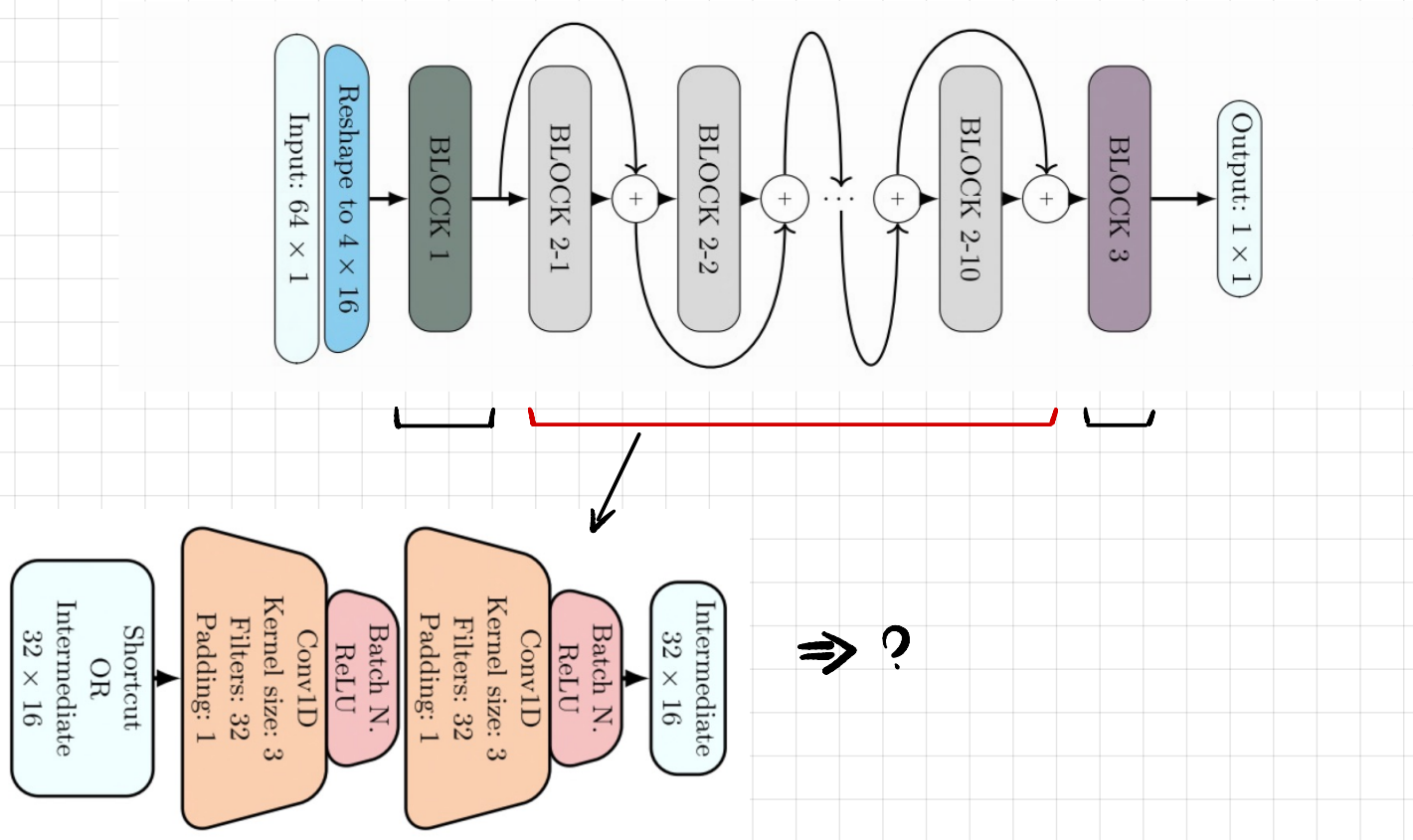



Fig. 4: The residual block (Blocks 2-i).

## Output Distribution Table (ODT)

DDT $\Delta C$

ODT $(\Delta L, \Delta V, V_0, V_1) \longrightarrow$ a generalization of DDT

## Masked Output Distribution Table (M-ODT) a compressed ODT

**Algorithm 3** Function used to construct the masked-output distribution table (M-ODT) from a dataset and a set of relevant masks.

**function** BUILD M-ODT($\mathcal{D}, R_M$)
  M-ODT = {}
  $n = |\mathcal{D}|$
  $P(\texttt{Real}) = 0.5$
  $P(\texttt{Random}) = 0.5$
  **for all** $M$ in $R_M$ **do**
    $N_h(M)$ = number of 1 in $M$
    $\mathcal{D}_M = \mathcal{D} \wedge M$ ▷ $\forall I \in \mathcal{D}, I_M = I \wedge M$
    U = Unique($\mathcal{D}_M$) ▷ Return a dictionary with key the element of $D_M$ and with value the number occurrence of that element in $D_M$
    **for all** $I_M$ in $\mathcal{D}_M$ **do**
      $P(I_M|\texttt{Random}) = 2^{-N_h(M)}$
      $P(I_M|\texttt{Real}) = \frac{1}{n} \times U[I_M]$
      $P(\texttt{Real}|I_M) = \frac{P(\texttt{Real})P(I_M|\texttt{Real})}{P(I_M|\texttt{Real})P(\texttt{Real})+P(I_M|\texttt{Random})P(\texttt{Random})}$
      M-ODT$[M][I_M] = P(\texttt{Real}|I_M)$
    **end for**
  **end for**
  **return** M-ODT
**end function**

→ extract from DNN (sort by score, apply local interpretation methods)

?

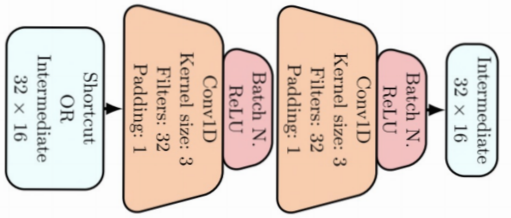



Fig. 4: The residual block (Blocks 2-i).

512

$$F = \begin{bmatrix} P(\text{real}|I_{M_1}) \\ P(\text{real}|I_{M_2}) \\ \cdots \\ P(\text{real}|I_{M_{|R_M|}}) \end{bmatrix}$$

Block 2-i ⇒ (1) compute ODT for ($\Delta L, \Delta V, V_0, V_1$)

(2) find several masks

(3) use masks to compress ODT to M-ODT

## Approximating the Expression of the Property $P$

(1) $(C, C') \rightarrow (\Delta L, \Delta V, V_0, V_1)$

(2) 512-vector $\rightarrow F = \begin{bmatrix} P(real \mid I_{M_1}) \\ P(real \mid I_{M_2}) \\ \vdots \\ P(real \mid I_{M_m}) \end{bmatrix}_{m \times 1}$

(3) MLP $\rightarrow$ LGBM

## Implementation

(1) 1st dataset : extract masks from DNN

(2) 2nd dataset : construct M-ODT

(3) 3rd dataset : train the final classifier

# Conclusion

1. distinguishers rely on
   - (1) ciphertext pair difference
   - (2) internal state difference in rounds
2. AKR <=> DNN <=> M-ODT
3. DNN is **not** producing novel cryptanalysis attacks

   ⇓

   **but** more like optimizing the **information extraction** with the low-data constraints

   more distinguisher settings
   machine learning pipelines
   types of ciphers
   ...

   } => have a better understanding