NeuroGIFT : Using a Machine Learning Based Sat Solver for Cryptanalysis

Ling Sun[1], David Gerault[2], Adrien Benamira[2], and Thomas Peyrin[2]

[1] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, Shandong University, Jinan, 250100, China
[2] Temasek Laboratories, Nanyang Technological University, Singapore

Keywords: Machine learning · Neural nework · SAT · GIFT · Cryptanalysis

Abstract. A recent trend in machine learning is the implementation of machine learning based solvers, such as the sat solver NeuroSat. The main limitation of NeuroSat is its scaling to large problems. We conjecture that this lack of scaling is due to learning an all-purpose SAT solver, and that learning to solve specialized SAT problems instead should yield better results. In this article, we evaluate our hypothesis by training and testing NeuroSat on SAT problems for differential cryptanalysis on the block cipher GIFT, and present the resulting classifier NeuroGift. We show that on these highly structured problems, our models are able to perform orders of magnitude better than the original NeuroSat, potentially paving the way for the use of specialized solvers for cryptanalysis problems.

1 Introduction

In recent years, machine learning techniques have become prominent for solving a wide range of problems. Recently, a promising method to solve combinatorial problems using machine learning was proposed. In NeuroSat [SLB+18], the authors propose to train a machine learning to solve combinatorial problems expressed in the SAT formalism. Since its publication in 2018, the article gained a lot of traction, and started a very active (over 80 citations to this day) research area on how to develop solvers based on machine learning.

In the field of cryptanalysis, SAT solvers, as well as other paradigms, such as MILP and constraint programming, are frequently used to evaluate the security of a primitive [MWGP11] [GMS16]. In particular, one of the most prominent forms of cryptanalysis, differential cryptanalysis, requires solving a heavily combinatorial problem as a starting point to a key recovery attack. Namely, this preliminary phase requires finding good *differential paths*, *i.e.*, propagation patterns from a difference between two plaintexts to a difference between two ciphertexts, that occur with a good probability. Among other optimisation tools, SAT solvers have been successfully used for this task [KLT15] [MP13]. However, the SAT problems studied for cryptanalysis typically have way more variables and clauses

than those solvable by NeuroSat (for instance, the problems studied in this article have thousands of variables). In addition, the main limitation of NeuroSat is its training regime : the generation of the training set requires to repeatedly call an external solver, and becomes impractical when the number of variables goes over a few hundreds [AMW19]. Therefore, it is not possible to directly apply the original NeuroSat to cryptanalysis problems. On the other hand, our cryptanalysis problems are very structured, as opposed to the random problems for which NeuroSat is trained. Therefore, our hypothesis is that the neural network can learn from the structure of these problems, yielding a cryptanalysis-oriented specialised solver, rather than a general purpose SAT solver. Hence, the research question we are interested in answering is

Can NeuroSat learn to solve highly specialized cryptanalysis problems more efficiently than generic random problems?

In this article, we present the experiments we led to solve that question. Training a neural network requires a training set composed of positive samples and negative samples. In our case, the positive samples are the SAT problems where the fixed input and output difference correspond to optimal differential characteristics (i.e., characteristics that have the minimal number of active S-boxes given a number of rounds), and the negative samples are SAT problems with fixed input and output differences that do no not correspond to optimal characteristic (i.e., for which the best possible characteristic has more active S-boxes than the overall optimal characteristic). We therefore need to be able to determine rapidly, for a large number of samples, the best possible characteristic given an input and output difference. We chose to perform our experiments on the block cipher GIFT, for which this task can be solved rapidly using CryptoMiniSat [SNC09] (less than 3 seconds per problem for 10 rounds).

On the problems we studied, our classifier, NeuroGift, showed remarkable performance. It was able to solve instances with significantly more variables than the original NeuroSat, obtain better accuracies, and generalise to bigger instances much better. Table 1 shows a comparison between the results obtained in the NeuroSat article and our best classifiers.

Table 1: Comparison of NeuroSat and NeuroGift

	NeuroSat	**NeuroGift**
Variables	$10 \leq n \leq 40$	$699 \leq n \leq 1494$
Training set	"Millions of pairs"	600 pairs
Best test accuracy	85%	100%

In this article, we present our experimental results, which can be summarised by the following contributions:

– We introduce NeuroGift, an adaptation of NeuroSat to the problem of determining whether there exists an optimal differential path for GIFT-64-128, satisfying a given input difference, output difference, and number of rounds.
– We present experiments aiming at validating that NeuroGift is actually able to learn to solve the corresponding SAT problem, rather than exploiting side information to do its classification.

2 Preliminaries

Since we aim to apply NeuroSAT to a specific kind of SAT problems in cryptanalysis, we first introduce the related SAT problems. Following that, some relevant information in the field of machine learning is provided. Finally, we recall NeuroSAT, which is the network we use in this paper.

2.1 GIFT-64-128

GIFT [BPP+17] is a family of lightweight block ciphers proposed at CHES 2017. As an improved version of PRESENT [BKL+07], it provides much-increased efficiency in all domains. At the same time, the well-known weakness of PRESENT regarding linear hull effect is overcome. There are two versions of GIFT - GIFT-64-128 and GIFT-128-128. We only focus on GIFT-64-128 in this paper and sometimes denote it as GIFT for short.

GIFT-64-128 is a 28-round Substitution Permutation Network (SPN) block cipher with 64-bit block size that supports 128-bit key. The round function, which is depicted in Figure 1, consists of standard operations such as substitution, permutation and subkey XOR. At the beginning of each round, 16 identical 4-bit S-boxes are applied in parallel as a non-linear substitution layer. Just after the substitution, a linear permutation is performed to provide diffusion, and finally, the state is XORed with the round key and the round constant. For more information, please find [BPP+17].

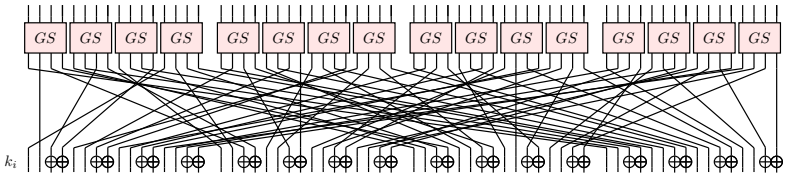


Fig. 1: Round function of GIFT-64-128.

2.2 Differential Cryptanalysis

Differential cryptanalysis was first introduced by Biham and Shamir in [BS90]. It is one of the most widely used and efficient forms of cryptanalysis. Consider

a function $f : \mathbb{F}_2^b \to \mathbb{F}_2^b$. Let x and x' be two different inputs for the function f with a difference $\Delta x := x \oplus x'$, and let $y = f(x)$ and $y' = f(x')$, such that $\Delta y = y \oplus y'$. Now, we are interested in the probability p such that $\Delta x \xrightarrow{f} \Delta y$. This can be calculated as such:

$$\mathbb{P}(\Delta x \xrightarrow{f} \Delta y) := \frac{\#\{x | f(x) \oplus f(x \oplus \Delta x) = \Delta y\}}{2^b} \tag{1}$$

If f is linear, then $\mathbb{P}(\Delta x \xrightarrow{f} \Delta y)$ can only be 0 or 1. Thus, the interest lies in when f is non-linear. In particular, for GIFT, the non-linear component is the S-box, which operates on consecutive nibbles (4-bit words) of the state. In order to study the propagation of a given difference through the S-box, the classical method is to build a Difference Distribution Table (DDT). Each entry of the DDT is of the form $(\Delta_{in}, \Delta_{out}, p)$, where p is the probability that the difference Δ_{in} results in the difference Δ_{out} after an S-Box.

The first step of differential cryptanalysis is to build *differential characteristics*, *i.e.*, difference propagation paths through the cipher. In particular, we are interested in maximizing the probability, over all plaintexts and differences,

$$\mathbb{P}^{opt}_{P,\delta_{in},\delta_{out} \in \mathbb{P}^3}(E(P) \oplus E(P') = \delta_{out} | P \oplus P' = \delta_{in})$$

Finding such an optimal path is a highly combinatorial problem. One of the common approaches to tackle it is to use SAT solvers.

2.3 SAT Problem

The boolean satisfiability problem (SAT) focuses on the satisfiability of a given Boolean formula. The SAT problem is *satisfiable* if the variables can be replaced with the values `True` or `False` so that the formula is evaluated to be `True`. It was shown that the problem is NP-complete [Coo71]. However, modern SAT solvers based on backtracking search can solve problems of practical interest with millions of variables and clauses [VHLP08].

For every Boolean formula, there is an equi-satisfiable formula in Conjunctive Normal Form (CNF), expressed as the conjunction ($\wedge$) of the disjunction ($\vee$) of (possibly negated) variables. Every conjunct of the Boolean formula in CNF is called a *clause*, and each (possibly negated) variable within a clause is called a *literal*. Since most SAT solvers regard problems in CNF as standard input, we are required to transform the question into an equivalent one in CNF when we plan to exploit SAT solver to solve it.

2.4 SAT Problems for `GIFT-64-128`

The problem we are interested in is about the optimal differential characteristic of `GIFT` with the minimum number of active S-boxes. An S-box is said to be active if it has a non-zero input difference. In practice, the number of active S-boxes provide a bound on the probability $\mathbb{P}^{opt}$, and is easier to obtain than

the exact probability of a characteristic. We adopt the method in [SWW18] to construct the corresponding SAT problems.

Basically, the clauses in the SAT problem can be divided into two groups. One group is used to propagate the input difference through the internal components of the cipher, and the other one depicts the objective function.

To trace the difference propagation of `GIFT`, the critical point lies in the manipulation of the S-box, since it is the unique non-linear operation inside the cipher. Denote $x_0\|x_1\|x_2\|x_3$ and $y_0\|y_1\|y_2\|y_3$ the input and output differences of the S-box, respectively. An auxiliary boolean variable w is introduced for each S-box to indicate whether the S-box is active or not. We aim to generate a group of clauses about x_i, y_i and w, and all the solutions of these clauses have a one-to-one correspondence with the elements in the following set

$$S = \left\{ \boldsymbol{x}\|\boldsymbol{y}\|w \;\middle|\; \begin{array}{c} \boldsymbol{x} \to \boldsymbol{y} \text{ is a possible propagation,} \\ w = x_0 \vee x_1 \vee x_2 \vee x_3 \end{array} \right\},$$

where $\boldsymbol{x} = x_0\|x_1\|x_2\|x_3$, $\boldsymbol{y} = y_0\|y_1\|y_2\|y_3$. The idea is to add clauses, which delete the vectors not belonging to the set S. To realise this goal, we first define a 9-bit boolean function

$$f(\boldsymbol{x}\|\boldsymbol{y}\|w) = \begin{cases} 1, \text{ if } \boldsymbol{x}\|\boldsymbol{y}\|w \in S \\ 0, \text{ else} \end{cases}.$$

According to the difference distribution table (DDT) of the S-box, we can generate the product-of-sum representation of f. Each term of the representation stands for a clause that deletes an impossible case in $\mathbb{F}_2^9 \setminus S$. This representation can be simplified by invoking Logic Friday[3] software. After that, from the simplified representation of f, the clauses tracing the differential propagation of the S-box are decoded. In total, we obtain 36 clauses, which can be found in Appendix A.

Note that the active S-boxes satisfy $w = 1$. The sum of w_i's $\sum_i w_i$ equals the number of active S-boxes of the characteristic. Since we target characteristics with the minimum number of S-boxes, the objective function is set as $\sum_i w_i \leqslant \tau$, where τ is a predetermined threshold. This kind of constraint is called cardinality constraint which can be transformed into a SAT problem in CNF with the sequential encoding method [Sin05]. Specifically, for the constraint $\sum_{i=0}^{n} w_i \leqslant \tau$, new dummy variables $u_{i,j}$ $(0 \leqslant i \leqslant n-2, 0 \leqslant j \leqslant \tau - 1)$ are introduced, and the

[3] http://sontrak.com/

following clauses will return unsatisfiable when $\sum_{i=0}^{n} w_i$ is larger than τ,

$$\begin{cases} \overline{w_0} \vee u_{0,0} = 1 \\ \overline{u_{0,j}} = 1 \\ \overline{w_i} \vee u_{i,0} = 1 \\ \overline{u_{i-1,0}} \vee u_{i,0} = 1 \\ \overline{w_i} \vee \overline{u_{i-1,j-1}} \vee u_{i,j} = 1 \\ \overline{u_{i-1,j}} \vee u_{i,j} = 1 \\ \overline{w_i} \vee \overline{u_{i-1,\tau-1}} = 1 \\ \overline{w_{n-1}} \vee \overline{u_{n-2,\tau-1}} = 1 \end{cases}, \tag{2}$$

where $1 \leqslant i \leqslant n-2$, $1 \leqslant j \leqslant \tau - 1$.

Denote u_i's the partial sums $u_i = \sum_{j=1}^{i} w_j$ for increasing the value of i up to the final $i = n$. The dummy variable $u_{i,j}$ denotes the j-th digit of the i-th partial sum u_i. With sequential encoding method, the constraint $\sum_{i=0}^{n} w_i \leqslant \tau$ is converted into Equation (2), and it is satisfiable only when the inequality constraint holds.

For the constraint $\sum_{i=0}^{n} w_i = \tau$, we only need to notice

$$\sum_{i=0}^{n} w_i = \tau \Leftrightarrow \sum_{i=0}^{n} w_i \leqslant \tau \text{ and } \sum_{i=0}^{n} \overline{w_i} \leqslant n - \tau.$$

That is, the equality constraint can be replaced with two sets of clauses, which are obtained by slightly adjusting the parameters in (2).

2.5 NeuroSAT

NeuroSAT [SLB+18] is designed as a neural classifier to predict the satisfiability of a SAT problem. In this section, we give an overview of NeuroSAT. For a more complete explanation and examples, please refer to [SLB+18]. The SAT problems are encoded as undirected graphs, and NeuroSAT operates on graphs as a Message Passing Neural Network (MPNN) [GSR+17]. Denote P a SAT problem with n literals composed of m clauses. The graph G_P of P consists of $2n + m$ nodes. Each element of the n pairs of complementary literals (x_i and $\overline{x_i}$) is represented as a node. Besides, one node is distributed for each of the m clauses. The edge between the literal x_i and the clause c_j exists if and only if c_j is related to x_i. To clarify the relationship between x_i and $\overline{x_i}$, a different type of line is allocated between the corresponding nodes. We define the characteristic function $\phi(P)$ of P as

$$\phi(P) = \begin{cases} 1, \text{ if } P \text{ is satisfiable,} \\ 0, \text{ otherwise.} \end{cases}$$

NeuroSAT acts as an approximation of the function $\phi(P)$.

In the message passing phase, the graph is embedded in a d-dimensional space. Each node has an embedding at each time step, and NeuroSAT iteratively updates this vector space embedding by passing messages back and forth along the edges of the graph. Denote $L^{(t)}$ a $2n \times d$ matrix where the i-th row stands for the embedding of the i-th literal l_i at the time step t. Let $C^{(t)}$ be an $m \times d$ matrix, and its j-th row represents the embedding of the j-th clause c_j at the time step t. The elements of $L^{(0)}$ and $C^{(0)}$ are initialised with a normal distribution. Let M be a $2n \times m$ matrix, which maintains the messages about the edges in the graph. $M(i,j) = 1$ if the literal l_i occurs in the clause c_j, otherwise, $M(i,j) = 0$. For encompassing the negation invariance of the SAT problem into the model, an operation F is introduced. F is parameterised by a matrix $L \in \mathbb{R}^{2n \times d}$, which contains the embeddings of all literals. The function of F is to swap the row corresponding to the embedding of x_i with the row of $\overline{x_i}$. An iteration consists of two stages:

1. each clause receives messages from all its neighbour literals and updates its embedding, accordingly;
2. each literal refines its embedding according to the messages from its neighbour clauses as well as its complementary literal.

These operations are implemented by two vanilla neural networks ($\mathbf{L_{msg}}$, $\mathbf{C_{msg}}$) and two LSTMs [HS97] ($\mathbf{L_u}$, $\mathbf{C_u}$). Formally, a single iteration can be expressed as

$$\left(C^{(t+1)}, C_h^{(t+1)}\right) \leftarrow \mathbf{C_u}\left(C^{(t)} \middle\| M^T \cdot \mathbf{L_{msg}}\big(L^{(t)}\big), C_h^{(t)}\right),$$
$$\left(L^{(t+1)}, L_h^{(t+1)}\right) \leftarrow \mathbf{L_u}\left(L^{(t)} \middle\| F(L^{(t)}) \middle\| M \cdot \mathbf{C_{msg}}\big(C^{(t+1)}\big), L_h^{(t)}\right),$$

where $L_h^{(t)} \in \mathbb{R}^{2n \times d}$ and $C_h^{(t)} \in \mathbb{R}^{m \times d}$ are the hidden states of $\mathbf{L_u}$ and $\mathbf{C_u}$, respectively. Please refer to Figure 2 for the framework of the iteration.

After T iterations, in the readout phase, a real number $y^{(T)}$ is computed as

$$L_*^{(T)} \leftarrow \mathbf{L_{vote}}\big(L^{(T)}\big),$$
$$y^{(T)} \leftarrow mean\big(L_*^{(T)}\big),$$

where $\mathbf{L_{vote}}$ is a 3-layer neural network, $L_*^{(T)}$ is a $2n$-dimensional vector. Denote the prediction of NeuroSAT for the problem P as $\texttt{NeuroSAT}(P)$. $\texttt{NeuroSAT}(P)$ depends on the value of $y^{(T)}$,

$$\texttt{NeuroSAT}(P) = \begin{cases} 1, \text{ if } y^{(T)} > 0, \\ 0, \text{ otherwise.} \end{cases}$$

NeuroSAT is trained to minimise the sigmoid cross entropy between $y^{(T)}$ and the correct label $\phi(P)$.

The training set of NeuroSAT is composed of pairs of random SAT problems on n variables. One problem in the pair is satisfiable, and the other one is unsatisfiable. The two samples differ by negating only a single literal occurring in

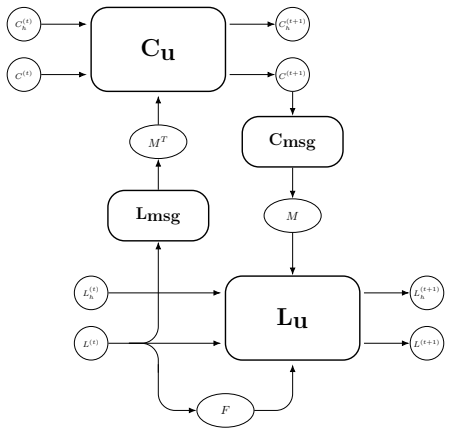


Fig. 2: An iteration of NeuroSAT.

one clause. All the pairs satisfying these properties constitute the set $\mathbf{SR}(n)$. In [SLB+18], the authors trained NeuroSAT with samples randomly drawn from $\mathbf{SR}(\mathbf{U}(10,40)) = \bigcup_{n=10}^{40} \mathbf{SR}(n)$ and tested it on $\mathbf{SR}(40)$. On average, the accuracy of NeuroSAT reaches 85%. When the network is generalised to $\mathbf{SR}(200)$, NeuroSAT can solve about 25% of them by running for more iterations of message passing [SLB+18].

3 NeuroGIFT

It was pointed in the paper [SLB+18] that the performance of the network on problems with more variables (e.g., $\mathbf{SR}(200)$) is not very good. However, in cryptanalysis, we are faced with problems with more than 200 variables. Table 2 lists the parameters of SAT problems for `GIFT` regarding different lengths. Our SAT problems for GIFT, from 2 rounds onwards, already have more than 200 variables. Therefore, in this section, we analyse the feasibility of training NeuroSAT on GIFT-related SAT problems. Then, the construction of the training set is introduced.

Table 2: Parameters of SAT problems for `GIFT`.

Round	1	2	3	4	5	6	7	8	9	10
#{Variables}	159	286	445	699	1017	1494	2067	2736	3358	4044
#{Clauses}	748	1433	2182	3120	4186	5569	7144	8911	10585	12387
#{Nodes}	1066	2005	3072	4518	6220	8557	11278	14383	17301	20475

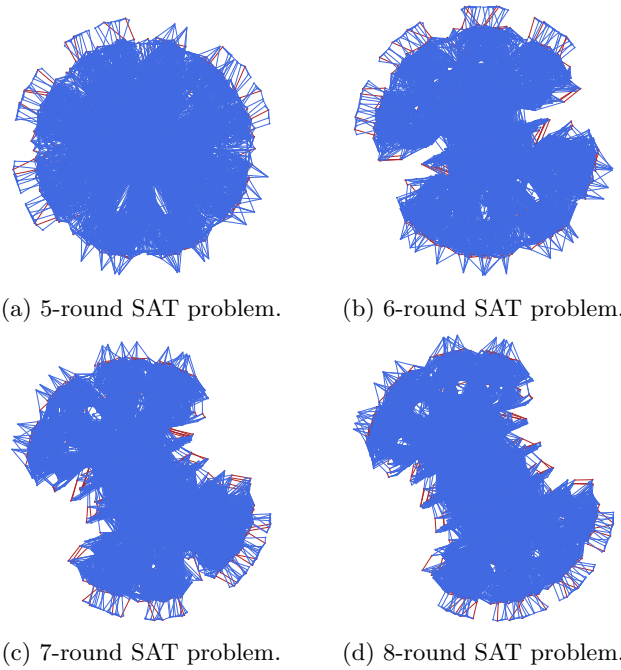
3.1 Motivations

Constricted domain of definition If there is no limitation on the system memory, the training set of NeuroSAT can be randomly drawn from $\mathbf{SR}(\mathbf{U}(1,\infty))$, which contains all possible SAT problems in theory. The ultimate goal of NeuroSAT is to approximate $\phi(P)$ by transforming P into graphs. Note that there is a one-to-one correspondence between $\mathbf{SR}(\mathbf{U}(1,\infty))$ and the set $\mathcal{G}$ of all graphs. NeuroSAT is trained to identify various features of all graphs. However, when we invoke SAT solvers to realise the automatic search of characteristics used in cryptanalysis, the SAT problems we are interested in constitute only a small sub-class of all possible SAT problems. That is to say, we do not require a powerful classifier as NeuroSAT. What we need in this case is a relatively weaker classifier, which only works well on a small sub-class of $\mathbf{SR}(\mathbf{U}(1,\infty))$. It is easier to train a customised classifier on a restricted domain, intuitively, since the features of the graphs in the sub-class are not as versatile as those in $\mathcal{G}$. Enlightened by this observation, we manage to apply NeuroSAT to identify the optimal differential characteristic of `GIFT`. We name this customised classifier as NeuroGIFT.

Similar structures in graphs Although NeuroSAT is trained with small-scale SAT problems, the authors attempt to extend its scope of application and employ it to solve bigger problems in the test phase. The performance of the network in the generalised case is not very good. A possible explanation is that the diversity of graphs is affected by the number of nodes in the graphs. Thus, in the generalised case, there may exist some features that NeuroSAT never saw during the training phase, and NeuroSAT does not know how to make decisions with these novel features. In cryptanalysis, when we consider the search of differential characteristics for iterative ciphers, the clauses for one round of difference propagation are iterated several times. Thus, the graphs of SAT problems regarding different lengths may share a similar structure. We illustrate the graphs of SAT problems from 5 rounds to 8 rounds of `GIFT` in Figure 3. From Figure 3, we can identify an apparent iterative property. The outer layers of these figures are similar, and the graphs corresponding to the long characteristics contain more internal layers than those related to short characteristics. This figure is to be compared with Figure 5, which shows that the graphs for lower number of rounds appear to be disconnected. In contrast, for 5 to 8 rounds, the graph is connected. We consider the possibility to apply a network trained with SAT problems no more than r rounds to predict the satisfiability of SAT problems longer than r rounds. The intuition is that if NeuroGIFT could learn the rule of iteration in the graphs, the generalisation would be more accessible than the case in NeuroSAT.

3.2 Construction of Training Set

In NeuroSAT, the information on the graph is involved in the matrix M. Thus, for NeuroSAT, identifying the graph is equivalent to recognising the matrix. The construction of $\mathbf{SR}(\mathbf{U}(1,\infty))$ ensures that M may take any pattern in theory. Nevertheless, since we restrict ourselves to the specific kind of SAT problems



(a) 5-round SAT problem. (b) 6-round SAT problem.

(c) 7-round SAT problem. (d) 8-round SAT problem.

Fig. 3: Graphs of our SAT problems with different numbers of rounds.

in NeuroGIFT, the pattern of M is almost fixed. In the case of `GIFT`, only the input/output differences and the number of rounds constitute the variations of M. To guarantee the generality of the model, we must generate the samples, carefully.

Similarly to the case of NeuroSAT, the training set of NeuroGIFT is composed of pairs of SAT problems - one is `SAT` sample, and the other one is `UNSAT` sample. Let k_r denote the number of active S-boxes in the optimal differential characteristic for r rounds. In all our SAT formulations, we add a constraint stating that the number of active s-boxes must be k_r. In our `SAT` samples, the variables of the SAT problem corresponding to the input and output difference are fixed to values $\delta_{in}, \delta_{out}$ such that there exists a differential characteristic with k_r active S-boxes starting with δ_{in} and ending with δ_{out}. In our `UNSAT` samples, the corresponding variables are set to $\delta_{in}, \delta_{out}$, such that there exists no differential trail with k_r active s-boxes starting with δ_{in} and ending with δ_{out}.

To generate the `SAT` samples in the training set, we first generate the input and output differences of all optimal characteristics for the corresponding number of rounds with the SAT solver Cryptominisat5[4]. Then, we randomly pick one pair of input and output differences and set them as the input and output differences of one `SAT` sample. In this manner, the `SAT` samples are created one by one.

[4] https://github.com/msoos/cryptominisat

The selection of the UNSAT sample is technical and will affect the quality of the final classifier. First, note that the complementary set of the set $\mathcal{O}$ consisting all optimal characteristics contains all possible characteristics overriding the condition of the minimum number of active S-boxes as well as all impossible characteristics, i.e., we have

$$\mathcal{U}\backslash\mathcal{O} = \mathcal{P} \cup \mathcal{I},$$

where $\mathcal{U}$ is the set of all characteristics, $\mathcal{P}$ denotes the set of all possible characteristics overriding the condition of the minimum number of active S-boxes and $\mathcal{I}$ stands for the set composed of all impossible characteristics. A natural way to draw the input and output differences of the UNSAT samples is to set them as random numbers. Because every bit of a random number has an equal chance of being a zero or a one, the Hamming weight of the random number in nibble is usually high[5]. However, the input and output differences of an optimal characteristic usually have relatively low Hamming weights. This distinction on the Hamming weight results in that the characteristics with random input and output differences only cover the cases in $\mathcal{I}$ and a subset of $\mathcal{P}$. A shortcoming of employing this kind of UNSAT samples is that NeuroSAT cannot learn comprehensive information in the underlying space of the training set. Just deciding by observing the Hamming weights of the input/output differences enables it to acquire high success probability in the training phase. Whereas it barely makes right predictions when we feed it with UNSAT samples having low Hamming weights in the input/output differences during the test phase.

To overcome this shortage, we should make sure that the UNSAT samples will adequately cover all cases in the set $\mathcal{U}\backslash\mathcal{O}$. We utilise the following procedures to generate the r-round UNSAT samples.

1. Suppose that the optimal r-round characteristic has k_r active S-boxes. We call Cryptominisat5 to output characteristics with $k_r + 1$, $k_r + 2$, ..., $16 \cdot r$ active S-boxes as many as possible[6] and store these solutions into File_{k_r+1}, File_{k_r+2}, ..., $\mathsf{File}_{16\cdot r}$, respectively.
2. Every time we are required to generate a UNSAT sample, we randomly select an integer seed s at random and compute the value $c = s \mod (16r + 1)$. If $k_r < c \leqslant 16r$, we sample a pair of input and output differences from the File_c and set them as the input and output differences of the UNSAT sample. Otherwise, the UNSAT sample is given with random input and output differences.

In this way, we guarantee that the UNSAT samples are almost uniformly distributed over the set $\mathcal{U}\backslash\mathcal{O}$. In the training phase, the network may 'see' different kinds of counterexamples, which include not only the characteristics with contradictions but also characteristics with a different number of active S-boxes. It

[5] The probability that any nibble of a random number equals 0x0 is 1/16.

[6] Since the solver has limited computation power, we cannot obtain all solutions. However, with the observation on the outputs, we think these solutions are enough to ensure the versatile of the sample space.

tries to learn the features in these graphs, and evaluate its learning outcome in the test phase.

Note that we do not take into account the differential effect: there may exist differentials for which the best differential characteristic has a relatively low probability, but that over all possible differential characteristics, have a high probability. However, we verify (using CryptoMiniSat), for each of our UNSAT samples, that the best corresponding differential characteristic is not optimal.

We present the three versions of the corresponding classifier, NeuroGift.

3.3 Three Versions of NeuroGIFT

Our experiments resulted in three versions of NeuroGift :

- NeuroGift-V1 is the baseline model. The samples are generated as described in Sect. 3.2.
- NeuroGift-trunc is designed to verify whether NeuroGift-V1 actually learns the resolution of the SAT problems. Hence, the variables corresponding to the objective function are removed from the SAT problems.
- NeuroGift-V2 is our best classifier. We keep the input and output differences in the `SAT` and `UNSAT` samples of one pair have same number of non-zero 4-bit nibbles. By forcing the SAT and UNSAT samples to be more similar, we hope to force NeuroSat to learn the actual resulution of the formula.

3.4 Parameter Setting

After each epoch of training, we evaluate the performance of the classifier on the training set. Let `T` denote a classification as positive, `F` denote a classification as negative, and let $X \in \{T, F\}, Y \in \{T, F\}$ respectively represent the prediction made by the classifier, and the ground truth. For instance, `TT` denotes number of samples the classifier correctly classified as positive, whereas `TF` denotes the number of samples classified as positive while actually being negative. The success probability (or accuracy) P_S of the classifier is

$$P_S = \frac{\mathtt{TT} + \mathtt{FF}}{\mathtt{TT} + \mathtt{TF} + \mathtt{FT} + \mathtt{FF}}.$$

This quantity expresses the fraction of the samples that are correctly classified. There are many tunable parameters for NeuroSat, which affect the performance of the network. It is observed that different parameters have different levels of influence on the model. We list those with non-negligible influences.

- The type of learning rate decay - There are three ways to modify the learning rate during the training phase, which are `no_decay`, `polynomial_decay` and `exponential_decay`. Usually, we are suggested to anneal the learning rate over time in training deep networks, since it may help us avoid wasting computation bouncing around chaotically with little improvement for a long time. However, when to decay the learning rate and how to decay it are somewhat difficult to determine because NeuroSAT is a very complicated network. So, we take a `no_decay` style in all experiments.

- Learning rate α - Adjusting the learning rate is a little technical. A high learning rate will make the system unstable, while it takes the model quite a long time to converge under a low learning rate.
- ℓ_2 weight - This term, which enables us to implement ℓ_2 regularisation, is used in the objective function. The intention of exploiting ℓ_2 regularisation is to escape overfitting and enhance the generalisation ability of the model.
- Clip value - It is used to clip the gradient, and this countermeasure allows us to ensure the gradient within a reasonable scale. With this method, we can effectively prevent the occurrence of gradient explosion, which is often encountered in training a deep network.

The memory complexity of NeuroSAT is related to the number of iterations T and the number of nodes in one batch. Increasing the number of iteration T, which is the depth of the deep network, improves the performance of NeuroSAT, potentially. Increasing the number of nodes in the batch will accelerate the training phase. In NeuroGIFT, we must allocate more nodes in the batch since the problems under consideration involve much more variables and clauses than those in NeuroSAT. Thus, the value of T remains unchanged in our case, that is, $T = 26$.

4 Experimental Results

All our experiments are performed with classifiers trained on problems between 1 and 6 rounds. In additional experiments, we evaluate the generalisation ability of these classifiers on problems from 7 to 9 rounds. We give the final accuracy of our classifiers with the following setting: The training and test set composed of 4 to 6 rounds samples. More specifically, we train the networks on 600 pairs of problems, composed of 200 4-round problems, 200 5-round problems, and 200 6-round problems. The test set is composed of 100 4-round problems, 100 5-round problems, and 100 6-round problems. We use a learning rate of 2×10^{-5}, ℓ_2 weight of 10^{-7} and clip value: 0.5.

4.1 NeuroGift-V1

Our first set of experiments directly applies the training method described in the previous section, and correspond to the classifier NeuroGift-V1.

In preliminary experiments, we use SAT problems varying from 1-round to 6-round to train and evaluate the model. We observe that, while the test results are good for problems from 1 to 3 rounds, they become heavily biased for the 4 to 6-rounds samples. For the 6-round samples, the network almost regards all `SAT` samples as `UNSAT` samples.

Our hypothesis is that the low-round samples have a negative effect on the accuracy of the resulting classifier. Indeed, the structure of the NeuroSat graph for these samples is different from the general structure for more rounds. This different structure is illustrated by Figure 5 (1- and 2-round graphs) and Figure 3

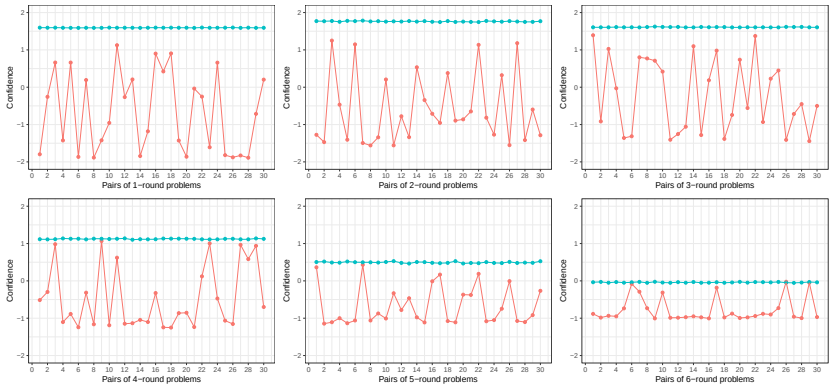
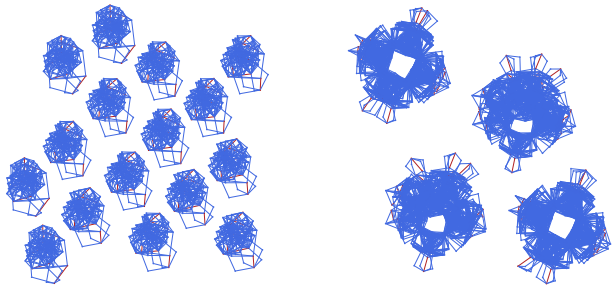


Fig. 4: Confidence of the NeuroGift-V1 trained with pairs from 1 to 6 rounds, in blue for the `SAT` samples, and in red for the `UNSAT` samples.

(5- to 8-round graphs). In particular, the graphs for shorter problems appear to be disconnected, as opposed to the graphs for larger problems (4 and more rounds). We conjecture that these disconnected graphs may lead the network to learn biased solving strategies.



(a) 1-round SAT problem. (b) 2-round SAT problem.

Fig. 5: Graphs corresponding to SAT problems with short lengths.

To verify our conjecture, we use a training set composed of problems varying from 4-round to 6-round to train the network. We evaluate the classifier on a test set with 60 pairs varying from 4-round problems to 6-round problems, and the levels of confidence of the network for these problems are shown in Figure 6. Note that this training set and the one used to train the classifier in Figure 6 have the same amount of samples, but the scale of the vertical axis is enlarged.

It therefore appears that, for the basic classifier NeuroGift-V1, the training set with problems on 4 to 6 rounds grants better results.

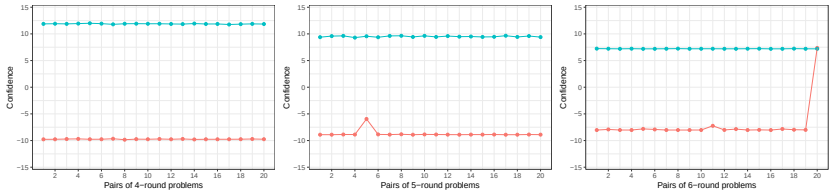


Fig. 6: Confidence of the model on the shrunken test set.

The final test accuracy, in the setting described at the beginning of the section, of the NeuroGift-V1 classifier, is 97%. From this classifier, we attempt to extract a satisfiable assignment from the SAT samples, following the methodology presented in the NeuroSat article. However, we were not able to extract a solution, leading us to wonder whether our model actually learns to solve the SAT problem. The corresponding experiments are presented in the next section.

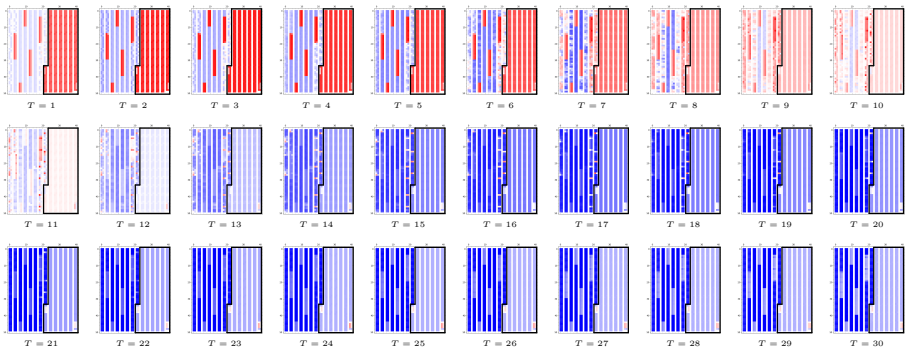
4.2 NeuroGift-trunc

One of our attempts at extracting a solution is illustrated on Figure 7, through the vectors $L_*^{(T)}$'s related to different T's ($1 \leqslant T \leqslant 30$). The positive values are represented in red while the negative values are displayed in blue. The darker the colour, the larger the absolute value of the number.

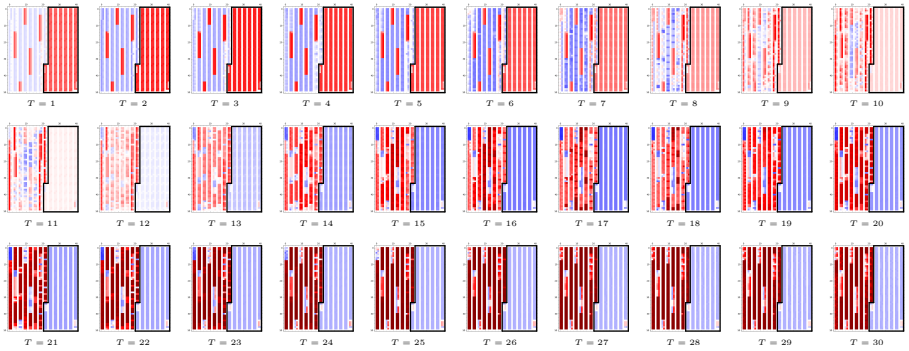
An unexpected phenomenon can be observed : the variables within the black frame, which are exactly the set of variables used to count the number of active SBoxes, seems to be irrelevant to the decision of the classifier. The same pattern can be observed for SAT and UNSAT samples.

Thus, we design NeuroGift-trunc to test whether NeuroGift-V1 really needs this part from the SAT problems. This version is different from NeuroGift-V1 in the construction of the samples. The setting of input and output differences for the `SAT` and `UNSAT` samples is the same as the case of NeuroGIFT-V1. However, for all samples, we delete the auxiliary variables and clauses corresponding to the objective function. Thus, SAT problem now encodes the question of whether the characteristic with the input and output differences is possible or not, whereas the labels are still related to the objective function.

The final accuracy of this model is 99%: NeuroGift-trunc performs even better than NeuroGift-V1. This is very counter-intuitive, as the UNSAT samples are not strictly unsatisfiable: the model does not really describe what a satisfiable sample is anymore. However, NeuroGift-trunc is able to predict the corresponding artificial labels. We conjecture that the very structure of the input and output difference may give enough information for the classifier to succeed solely based on the corresponding litterals. We therefore design a new model, NeuroGift-trunc, where the training set is more carefully designed to eliminate this structure.



(a) `UNSAT` sample.



(b) `SAT` sample.

Fig. 7: Propagation of the vector $L_*^{(T)}$.

NeuroGift-V2 In general, differential characteristics with an optimal number of active SBoxes are such that their input and output difference have a given structure. Typically, the number of non-zero nibbles in these differences is low. While our experiments, decribed in Appendix B, did not provide definitive evidence that NeuroGift-V1 makes decisions by counting the number of non-zero nibbles of the input/output differences, we still wonder its performance after removing this feature from the training set. In particular, making the number of non-zero nibbles similar for the SAT and UNSAT samples may force the classifier to learn more specialized resolution features. The resulting classifier is called NeuroGift-V2. The innovation lies in the construction of the `UNSAT` samples. In one pair of samples, we ensure that the Hamming weights of the input/output differences of the `UNSAT` and `SAT` sample are equal, which is accomplished by the following steps.

1. We randomly sample a pair of input and output differences corresponding to an r-round optimal characteristic. Then, the Hamming weight h_{in} of the input difference and the Hamming weight h_{out} of the output difference are computed, respectively.

2. Two sets of integers $\{p_0^{in}, p_1^{in}, \ldots, p_{h_{in}-1}^{in}\}$ and $\{p_0^{out}, p_1^{out}, \ldots, p_{h_{out}-1}^{out}\}$ satisfying the following conditions are generated:
   - p_i^{in} and p_j^{out} are random positive integers no more than 16;
   - $p_i^{in} \neq p_j^{in}$ for all $0 \leqslant i < j \leqslant h_{in} - 1$;
   - $p_i^{out} \neq p_j^{out}$ for all $0 \leqslant i < j \leqslant h_{out} - 1$.

   p_i^{in}'s and p_j^{out}'s point out the non-zero nibble positions in the input and output differences of the `UNSAT` sample.
3. The positions of the input (resp. output) difference lie in the set $\{p_0^{in}, p_1^{in}, \ldots, p_{h_{in}-1}^{in}\}$ (resp. $\{p_0^{out}, p_1^{out}, \ldots, p_{h_{out}-1}^{out}\}$) are set with random non-zero 4-bit values. Moreover, the remaining positions are fixed as `0x0`.

With this method, we eliminate the effect of the Hamming weight on the training set.

On the same test set as the other 2 variants, NeuroGift-V2 achieves 100% accuracy. However, we were still not able to extract a solution from the variable embeddings.

4.3 Generalisation to More Rounds

The results of the three models are consistent, and the best test accuracies we obtained are respectively 97%, 100% and 99%. For comparison, the results of the original NeuroSat article are given in Table 1. These results are encouraging, and seem to give a positive answer to our main question, which was to determine whether NeuroSat could perform better on sets of problems sharing similar structures, rather than random problems.

In essence, with only 600 pairs, our models were able to reach as much as 100% accuracy, whereas the best NeuroSat instance presented in the original article only reached 85% accuracy, despite being trained on millions of pairs, and studying problems with over 15 times less variables. For applications in cryptography, the ability of a neural network to make predictions for more rounds than it was trained for is very important. In particular, while solving the problem for a few rounds might be easy, it generally becomes exponentially harder as the number of rounds increases. Therefore, when applying the techniques of NeuroGift to other ciphers, we will not necessarily be able to generate an appropriate training set for a high number of rounds efficiently.

In order to evaluate the generalisation abilities of NeuroGift, we pick our best model (NeuroGift-V2), and tune its parameters for better results. After performing control experiments, the setting that resulted in the best generalisation was the following. We train the model on 500 pairs of problems from 1 to 5 rounds (100 of each), with learning rate 10^{-5}, l2 weight : 10^{-9}, and clip value : 0.5. Under this setting, we evaluated the generalisation of NeuroGift-V2 on a different test set for each number of rounds, from 6 to 10. The resulting test accuracies are given in Table 3.

The generalisation accuracy remains very close to 100%, even for 10 rounds problems, even though 10-rounds problems have 4 times more variables (4044)

Rounds	6	7	8	9	10
Accuracy	100%	100%	99%	99%	98%

Table 3: Generalisation ability of NeuroGIFT-V2.

compared to the 5-rounds samples seen in training. As a comparison, in the original NeuroSat article, the accuracy dropped to approximately 40% in a similar setting (going from at most 40 variables to 160), even though the number of message passing iterations was increased from 26 to 1000. In contrast, we restricted ourselves to 26 iterations. In additional experiments, we observed that NeuroGift-V1 and NeuroGift-trunc did not generalise as well as NeuroGift-V2. In particular, the accuracy of NeuroGift-trunc drops to below 90% for 9 rounds. This could be an indication that NeuroGift-V2 actually learns something closer to the actual resolution of the SAT problem, even though we were not able to completely confirm it.

5 Conclusion

Related Work and Extentions. Following the publication of NeuroSat, a wide range of articles proposing extentions were published. We believe the most promising one for our application is the PDP framework. The PDP framework [AMW19], which is an extension of the the CircuitSAT framework [AMW18], belongs to the deep learning SAT solver family as NeuroSAT [SLB+18]. But whereas NeuroSAT is a supervised framework, the work of Amizadeh et al. takes the advantage of the probabilistic inference formulation in order to propose an unsupervised setting. In fact, they introduce a differential expression of the energy function that they want to minimize. With this formulation of the problem, the work in [AMW19] outperforms the NeuroSAT model. Moreover, the work allows three different times for learning (Propagation Decimation and Prediction) which leads to an hybrid model. In fact, the three stages are modular: they can be a fully a neural embedding block or they can be replace by a traditional non trainable block (like the Survey-propagation guided decimation algorithm [MMM09] as propagator block for example). However, despite a highly parallelizable model, the training and the inference is quite long (in comparison to NeuroSAT) when the number of variable of the SAT problem growth. This is certainly due to the combination of the fact that the model has twice more embedding than NeuroSAT and the unsupervised setting. Finally, the second shortcoming of the model is that it is not clear how the model can label an UNSAT problem.

Discussion of our Results The use of specialized solvers based on machine learning, rather than classical solvers, seems to be a promising research direction. The reason why boils down to the distinction between genericity and speciality: a solver that only aims at solving cryptanalysis problem does not need to be

good at unrelated problems, and may therefore perfom better on very specific problems. The main limitation to the use of such solvers, if the scaling issues are solved, will be their approximate nature. A solver such as NeuroGift gives a likelyhood for the presence of a characteristic with k active S-boxes, as opposed to a traditional solver that would give an exact answer. While an approximation is, in itself, useful (after all, using the best differential characteristic is, in itself, an approximation to the resistance of a cipher against differential attacks), we believe further research should consider integrating machine learning solvers as heuristics to drive the search of classical solvers. This approach has proven efficient for generic SAT solving [SB19]. We believe combining machine learning based approaches with state-of-the-art solvers will enable progress on problems that are still difficult for classical solvers, such as cryptanalysis problems on hash functions.

Conclusion. In this article, we present models for the resolution of differential cryptanalysis problems with NeuroSat. We show that, when trained on a restricted set of problems, rather than the set of all SAT problems, the resulting classifier NeuroGift scales to significantly more variables than the original NeuroSat. However, more experiments are required to confirm that NeuroGift is able to determine the values of the variables, rather than just classifying based on some hidden structure in the input and output differences. In particular, future works includes the design of a model where the structure of the UNSAT samples is even closer to that of the SAT sample, in order to force NeuroGift to learn the actual resolution. For instance, we could set the input and output differences of the UNSAT samples to those of two different SAT samples.

While the results presented in this paper are encouraging, they do not adress a fundamental limitation of NeuroSat : the size of the generated graph. For 10 rounds, the graph already has over 20000 nodes. As a comparison, on our original benchmarking GPU (GTX 970), we were not able to generate the graph for more than 7 rounds (11278 nodes) without exhausting the graphic card's memory. Therefore, for a broader application of these methods on harder cryptanalysis problems, a solution must be found to restrict the size of the graph. The experiments performed with NeuroGift-trunc seem to be a promising option : for 10 rounds, the size of the graph is only 7616 nodes. On the other hand, the generalisation capabilities of NeuroGift-trunc are not on par with NeuroGift-V2, so further improvements are needed. A potentially promising alternative left for future work would be to merge NeuroGift-V2 and NeuroGift-trunc into a single model, with the training set constraints of NeuroGift-V2, and the truncation of the objective function from NeuroGift-trunc. Our hope is that the results presented in this paper lay the groundwork for a larger scale application of machine learning based solvers to cryptanalysis problems. From these first experimental results, future research directions include studying primitives and number of rounds which are more challenging to for classical dedicated solvers.

References

[AMW18] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Learning to solve circuit-sat: An unsupervised differentiable approach. 2018.

[AMW19] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. Pdp: A general neural framework for learning constraint satisfaction solvers. *arXiv preprint arXiv:1903.01969*, 2019.

[BKL+07] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[BPP+17] Subhadeep Banik, Sumit Kumar Pandey, Thomas Peyrin, Yu Sasaki, Siang Meng Sim, and Yosuke Todo. GIFT: A small present - towards reaching the limit of lightweight encryption. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 321–345. Springer, 2017.

[BS90] Eli Biham and Adi Shamir. Differential cryptanalysis of DES-like cryptosystems. In Alfred Menezes and Scott A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, 10th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1990, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 2–21. Springer, 1990.

[Coo71] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158. ACM, 1971.

[GMS16] David Gerault, Marine Minier, and Christine Solnon. Constraint programming models for chosen key differential cryptanalysis. In Michel Rueher, editor, *Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings*, volume 9892 of *Lecture Notes in Computer Science*, pages 584–601. Springer, 2016.

[GSR+17] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[KLT15] Stefan Kölbl, Gregor Leander, and Tyge Tiessen. Observations on the SIMON block cipher family. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 161–185. Springer, 2015.

[MMM09] Marc Mezard, Marc Mezard, and Andrea Montanari. *Information, physics, and computation*. Oxford University Press, 2009.

[MP13] Nicky Mouha and Bart Preneel. Towards finding optimal differential characteristics for arx: Application to salsa20. *Cryptology ePrint Archive, Report 2013/328*, 2013.

[MWGP11] Nicky Mouha, Qingju Wang, Dawu Gu, and Bart Preneel. Differential and linear cryptanalysis using mixed-integer linear programming. In *International Conference on Information Security and Cryptology*, pages 57–76. Springer, 2011.

[SB19] Daniel Selsam and Nikolaj Bjørner. Neurocore: Guiding high-performance SAT solvers with unsat-core predictions. *CoRR*, abs/1903.04671, 2019.

[Sin05] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005, 11th International Conference, CP 2005, Sitges, Spain, October 1-5, 2005, Proceedings*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[SLB+18] Daniel Selsam, Matthew Lamm, Benedikt Bunz, Percy Liang, Leonardo de Moura, and David L Dill. Learning a sat solver from single-bit supervision. *arXiv preprint arXiv:1802.03685*, 2018.

[SNC09] Mate Soos, Karsten Nohl, and Claude Castelluccia. Extending SAT solvers to cryptographic problems. In Oliver Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*, pages 244–257. Springer, 2009.

[SWW18] Ling Sun, Wei Wang, and Meiqin Wang. More accurate differential properties of LED64 and Midori64. *IACR Transactions on Symmetric Cryptology*, 2018(3):93–123, Sep. 2018.

[VHLP08] Frank Van Harmelen, Vladimir Lifschitz, and Bruce Porter. *Handbook of knowledge representation*. Elsevier, 2008.

Appendix

A Clauses for the S-box of GIFT

The 36 clauses are provided as follows

$$
\begin{cases}
\overline{x_0} \vee \overline{x_1} \vee \overline{x_2} \vee x_3 \vee \overline{y_2} = 1 \\
x_0 \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{y_1} \vee \overline{y_2} = 1 \\
x_0 \vee \overline{x_1} \vee x_2 \vee \overline{y_0} \vee y_1 \vee \overline{y_2} = 1 \\
x_1 \vee x_2 \vee \overline{y_0} \vee \overline{y_1} \vee \overline{y_2} \vee y_3 = 1 \\
x_1 \vee \overline{x_2} \vee \overline{x_3} \vee y_1 \vee \overline{y_2} \vee y_3 = 1 \\
x_0 \vee x_1 \vee x_2 \vee \overline{y_0} \vee \overline{y_1} \vee y_2 \vee \overline{y_3} = 1 \\
x_0 \vee x_1 \vee \overline{x_2} \vee \overline{x_3} \vee y_1 \vee y_2 \vee \overline{y_3} = 1 \\
\overline{y_0} \vee w = 1 \\
\overline{y_1} \vee w = 1 \\
\overline{y_2} \vee w = 1 \\
\overline{y_3} \vee w = 1 \\
x_0 \vee \overline{x_1} \vee \overline{x_2} \vee x_3 \vee y_2 = 1 \\
x_1 \vee \overline{x_2} \vee x_3 \vee \overline{y_2} \vee \overline{y_3} = 1 \\
\overline{x_1} \vee x_3 \vee \overline{y_0} \vee \overline{y_2} \vee \overline{y_3} = 1 \\
\overline{x_0} \vee \overline{x_1} \vee y_0 \vee \overline{y_2} \vee \overline{y_3} = 1 \\
x_0 \vee \overline{x_3} \vee y_0 \vee \overline{y_2} \vee \overline{y_3} = 1 \\
\overline{x_0} \vee x_2 \vee x_3 \vee y_2 \vee \overline{y_3} = 1 \\
\overline{x_0} \vee y_0 \vee \overline{y_1} \vee y_2 \vee \overline{y_3} = 1 \\
x_0 \vee \overline{x_1} \vee x_2 \vee x_3 \vee y_3 = 1 \\
\overline{x_0} \vee x_1 \vee x_2 \vee x_3 \vee y_3 = 1 \\
\overline{x_1} \vee y_0 \vee \overline{y_1} \vee \overline{y_2} \vee y_3 = 1 \\
x_1 \vee \overline{x_2} \vee x_3 \vee y_2 \vee y_3 = 1 \\
x_1 \vee \overline{x_3} \vee y_0 \vee y_2 \vee y_3 = 1 \\
x_0 \vee y_0 \vee \overline{y_1} \vee y_2 \vee y_3 = 1 \\
\overline{x_1} \vee y_0 \vee y_1 \vee y_2 \vee y_3 = 1 \\
x_0 \vee x_1 \vee x_2 \vee x_3 \vee \overline{w} = 1 \\
x_0 \vee y_0 \vee y_1 \vee y_2 \vee \overline{w} = 1 \\
x_1 \vee y_0 \vee y_1 \vee y_3 \vee \overline{w} = 1 \\
x_0 \vee \overline{x_1} \vee x_2 \vee \overline{x_3} \vee y_1 \vee \overline{y_3} = 1 \\
\overline{x_0} \vee x_1 \vee \overline{x_3} \vee \overline{y_0} \vee \overline{y_2} \vee \overline{y_3} = 1 \\
\overline{x_0} \vee x_2 \vee \overline{y_0} \vee y_1 \vee y_2 \vee \overline{y_3} = 1 \\
\overline{x_0} \vee \overline{x_1} \vee \overline{x_3} \vee \overline{y_0} \vee y_2 \vee y_3 = 1 \\
\overline{x_0} \vee x_1 \vee \overline{x_2} \vee \overline{x_3} \vee \overline{y_0} \vee \overline{y_1} \vee \overline{y_3} = 1 \\
\overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{y_0} \vee \overline{y_1} \vee y_2 \vee \overline{y_3} = 1 \\
\overline{x_0} \vee \overline{x_1} \vee x_2 \vee \overline{x_3} \vee \overline{y_0} \vee \overline{y_1} \vee y_3 = 1 \\
\overline{x_0} \vee \overline{x_1} \vee \overline{x_2} \vee \overline{x_3} \vee \overline{y_0} \vee y_1 \vee y_3 = 1
\end{cases}
$$

B Impact of the Samples with Low Hamming Weight

In order to evaluate the impact of samples where the difference has low hamming weight, we plot the confidence of the network after each message passing iteration during the test phase of NeuroGift-V1, while keeping track of the structure of the input and output difference. Figure 8 illustrates the confidence of the network for 6-rounds samples. We can observe that the curves for all `SAT` samples of the same length are similar. But the curves of the `UNSAT` samples are different. We think that two parameters result in the difference. One reason is the Hamming weight of the input/output differences. Another one is the differential effect. We analyse the 60 pairs of samples in the test set. The Hamming weight of the `UNSAT` samples are provided in the figures. The samples with the same Hamming weight are illstrated with same color. It can be found that the curves with same color are similar. An interesting example is the `UNSAT` sample of the 20-th pair of 6-round samples, this is the unique sample that is wrongfully classified by the network. Firstly, the Hamming weight of the input/output differences of this sample is 4, which is even smaller than the value of the optimal trail. Another fact is that the corresponding differential only have one trail with 16 active S-boxes. Since NeuroGift-V1 is designed to identify the topology structure of the figure, the dominant trail property causes the network to make the wrong decision.

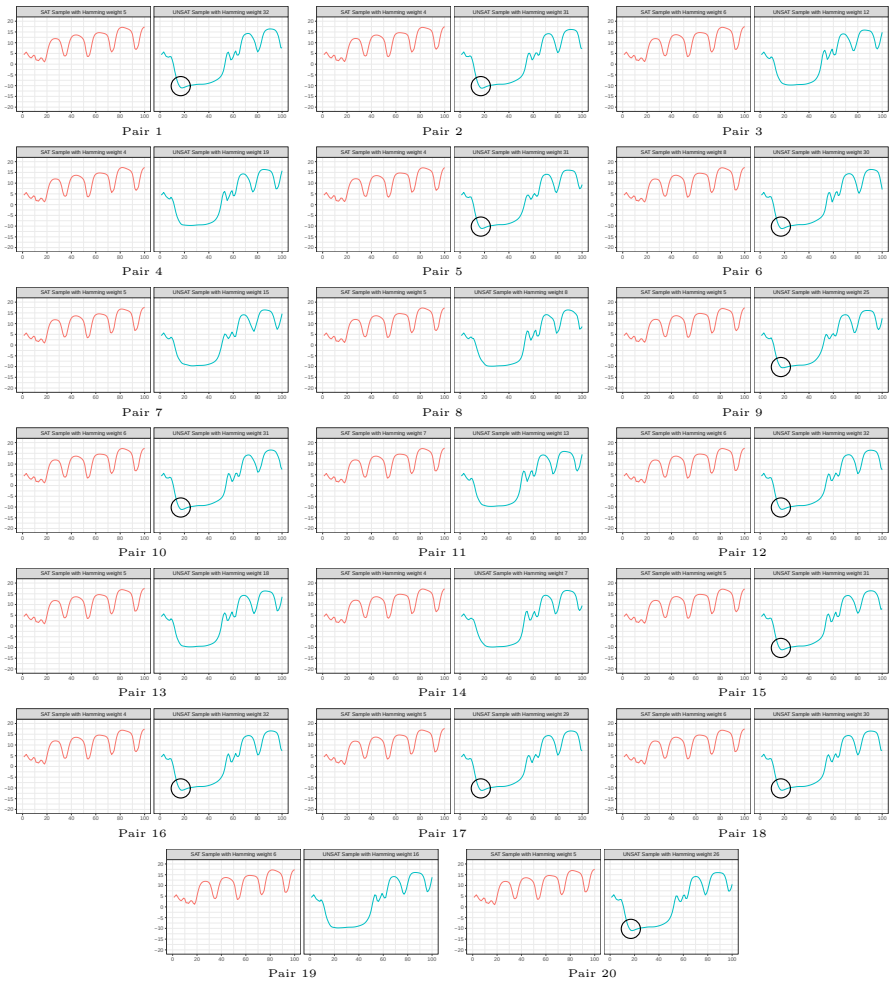


Fig. 8: Confidence for 6-round Samples. The horizontal axis corresponds to the number of iterations, and the vertical axis is the value of $y^{(T)}$. The circles show a characteristic feature of the `UNSAT` samples.