Learning with Enumeration: Neural-Guided SAT Framework for Cryptographic Key Recovery

Xinhao Zheng, Xinhao Song, Jinchen Yu, Zhuoyuan Xu, Gongshen Liu, Junchi Yan[‡]
School of Computer Science & School of AI, Shanghai Jiao Tong University
Shanghai Innovation Institute
{void_zxh,sxh001,yjc0719,andyxuzy,lgshen,yanjunchi}@sjtu.edu.cn

Abstract

Boolean satisfiability (SAT) provides a foundation tool for cryptographic key recovery by encoding ciphers into CNF or ANF representations. However, heuristic solvers and their neural-enhanced variants often degrade significantly on cryptographic instances due to pronounced structural symmetry, a large number of intermediate variables, and complex algebraic dependencies. Existing neural approaches—ranging from end-to-end prediction to solver-integrated heuristics—face challenges in scalability, assignment accuracy, or computational efficiency. In this paper, we propose a unified neural-guided enumerative SAT framework for cryptographic key recovery. It performs a single forward neural guidance to identify a subset of k critical variables, followed by enumeration over their assignments before invoking a heuristic SAT solver. This design effectively reduces the combinatorial search space while keeping neural overhead minimal. We further introduce a taxonomy of neural guidance across data and model capability regimes, including supervised selection with cryptographic priors, uncertainty-driven selection with assignment prediction, and robust fallback strategies using the model trained on general SAT datasets. Experiments on 10 SAT solvers and 2 cube-and-conquer strategies over SAT4CryptoBench and SAT Competition benchmarks demonstrate up to $5\times$ speedup and approximately $2\times$ average improvement on cryptographic instances, while maintaining better performance on general SAT datasets. These results highlight the effectiveness and generalization of this framework, illustrating its potential to bridge machine learning and SAT solving in cryptanalysis tasks.

1 Introduction

Boolean satisfiability (SAT) solvers provide a principled tool for cryptographic key recovery by encoding ciphers into conjunctive normal form (CNF) or algebraic normal form (ANF). Traditional heuristic solvers and their machine learning enhanced variants [12, 28, 29, 36] have demonstrated strong performance on general SAT instances. Recent neural approaches further explore end-to-end cryptanalysis and neural-assisted SAT solving. End-to-end methods such as NeuroSAT [31], CryptoANFNet [46], and neural differential distinguishers [13] learn structural patterns from CNF, ANF, or ciphertext representations, while neural-assisted SAT methods incorporate learning into branching heuristics [28, 29], variable selection mechanisms [11], and hyperparameter optimization [10].

However, cryptographic SAT instances remain fundamentally challenging due to strong structural symmetry, massive numbers of intermediate variables, and complex algebraic dependencies. As cipher complexity increases, the search space grows exponentially, leading to severe performance degradation for heuristic SAT solvers. Meanwhile, end-to-end neural models still exhibit limited assignment prediction capability and are restricted to small-scale instances. Moreover, even when

[‡]Correspondence author. Work was partly supported by the National Cyber Security-National Science and Technology Major Project (No.2026ZD1500700) and NSFC (92370201).

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

predictions are locally accurate, bit-level errors tend to accumulate exponentially. Neural-assisted heuristic SAT solvers also struggle to learn effective guidance on large cipher-derived formulas, while the integration of neural models introduces substantial computational overhead that limits scalability.

To address these challenges, we propose a unified neural-guided enumerative SAT framework, which integrates neural models with enumeration based on ***data prior and model capability of assignment prediction***. We use a neural-guided module to select k critical variables, which are then enumerated over all candidate assignments. The resulting CNF instances are subsequently passed to a heuristic SAT solver, effectively steering the search toward a reduced and more informative subspace. In this design, the neural model is invoked only once at the beginning, thereby substantially reducing computational overhead while maintaining solver effectiveness. Specifically, we further categorize neural guidance into three regimes according to different settings of data prior and model capability:

- **Background-Accessible Data & Supervision-Aligned Selection**: With explicit cryptographic background (e.g., plaintext–ciphertext pairs) and no requirement for assignment prediction, we train a classifier to select variables from a small candidate set, using the subset with the shortest solver runtime as supervision. Enumeration is then performed with fixed initial assignments.
- **Structure-Aligned Data & Uncertainty-Driven Selection**: With structure-aligned data (without background priors) and only limited assignment prediction capability, we train a model to predict variable assignments, identify uncertain variables (accuracy ≈ 0.5), and select those that are both uncertain and frequent in non-unit clauses for enumeration with fixed initial assignments.
- **Unstructured Data & Initial-Assignment Prediction**: With effective assignment prediction but unstructured data, we use a model trained on general SAT data to predict initial assignments, and select variables with high clause participation for enumeration as a robust fallback strategy.

We evaluate our framework on 10 heuristic SAT solvers and 2 cube-and-conquer strategies across both cryptographic instances from SAT4CryptoBench [47] and standard SAT Competition benchmarks. The results show that our framework consistently outperforms strong SAT solvers and their cube-and-conquer variants, achieving up to $5\times$ speedup and an average improvement of about $2\times$ over state-of-the-art solvers on cryptographic benchmarks under the first two regimes. In addition, it maintains better performance on general SAT benchmarks under the third regime, demonstrating strong generalization ability across different problem settings. These results reveal a unified insight: **Enumeration can act as an adapter between data priors and model capacity**, where a single neural step selects a suitable variable subset and assignment space, improving SAT solving efficiency with minimal overhead. **Overall, the contributions are:**

- We propose a unified neural-guided enumerative SAT framework for cryptographic key recovery, where a single neural step selects critical variables and enumeration is applied before heuristic SAT solving, substantially reducing search complexity with minimal overhead.
- We introduce a principled taxonomy of neural guidance across data and model regimes, including supervised selection with cryptographic priors, uncertainty-driven selection with assignment prediction, and robust fallback using the model trained on general SAT datasets.
- We demonstrate strong performance on cryptographic and general SAT benchmarks, achieving up to $5\times$ speedup over excellent SAT solvers in Table 1, while maintaining strong generalization, and show that enumeration effectively bridges data priors and model capacity.

2 Related Works

Differential Cryptanalysis with Neural Networks. Differential cryptanalysis, as an important technique in cryptanalysis, exploits the patterns of differences in plaintext pairs to infer information about the secret key. Ghor et al. [13] pioneered deep learning for differential cryptanalysis, showing that convolutional networks can distinguish correct from incorrect differential characteristics on reduced-round Speck32/64. Then, Hou et al. [16] extended attacks to Simon32/64 and other lightweight ciphers, while Benamira et al. [5] systematically evaluated their strengths and limitations. Subsequent works explored reinforcement learning for automated differential trail discovery [2], and improved training strategies [32, 35]. Recent studies introduced mixture-based or enhanced neural differential attacks [3, 43], and the work [34] comprehensively surveys these developments. However, this type of neural-discriminator-based method remains costly: they mainly filter promising differential trails or key subspaces, but full key recovery still requires exhaustive enumeration with a time complexity of near 2^{31}. The methods also struggle with single plaintext–ciphertext scenarios

and require large training datasets (over 10^7 instances). Besides, the accuracy at the key-recovery level remains limited, leaving a substantial gap compared to traditional SAT-based solvers.

Neural SAT-based Cryptanalysis. Algebraic cryptanalysis can be formulated as a satisfiability problem, motivating another type of neural-discriminator-based methods that apply neural networks to SAT solving for cryptography. Early approaches employed recursive neural networks to process CNF formulas as sequences, which later evolved into graph neural networks (GNNs) capable of directly capturing the structural information of formulas [8]. NeuroSAT [31] exemplifies this line of work, treating SAT solving as a prediction problem and using a neural network to predict satisfiability on CNF instances from single-bit supervision, thereby providing an end-to-end framework for SAT solving. Related efforts further integrated the solving process with neural models, including GNNs [9, 44] and transformer-based architectures [33], to tackle boolean constraints arising in SAT solving. Furthermore, Li et al. [21] proposed a dedicated benchmark to facilitate benchmarking and evaluation of GNN-based SAT solvers. Building on these developments, CryptoANFNet [46] extended the approach to cryptographic applications by training classifiers on ANF representations to distinguish correct from incorrect one-bit key assignments for key solving. However, these methods primarily focus on satisfiability with limited accuracy in key assignments, resulting in modest key-recovery performance and constrained practical applicability.

ML-based heuristic SAT Solvers. A complementary line of research augments classical SAT solvers with machine learning guidance, including neural variable heuristics [22], clause and restart policy learning [24], portfolio selection [45], and hyperparameter optimization [10, 41]. In the general SAT domain, these heuristic-solver-based methods help prune large regions of the search space or prioritize promising assignments, effectively serving as a form of guided enumeration. For example, neural heuristics can bias branching toward likely-satisfiable subspaces [17, 19], and learned clause deletion [39] or glue-variable prediction can reduce redundant exploration [14]. However, when applied to cryptographic SAT instances, these approaches face notable challenges. SAT4CryptoBench [47] showed that the integration of neural models frequently introduces significant computational overhead, limiting scalability and generalization. Moreover, cipher-derived SAT instances often involve many intermediate variables, making it difficult for learning heuristics.

To address the above challenges, we propose a unified neural-guided enumerative SAT framework that integrates neural-guided selection with enumeration under varying data priors and model capabilities. The framework reduces search space and solving time while preserving generalization, and enables strong performance in high-prior cryptographic instances and applicability to general SAT solving.

3 Preliminaries

CNF & ANF representation. Boolean functions are often represented in two standard forms: conjunctive normal form (CNF) and algebraic normal form (ANF). CNF expresses a Boolean formula as a conjunction of clauses, each being a disjunction of literals, which is the standard input format. For example, the CNF formula $(x_1 \vee \neg x_2) \wedge (x_2 \vee x_3 \vee \neg x_4)$ contains the clause $(x_1 \vee \neg x_2)$, which in a CNF file is written as `1 -2 0`, and $x_1, \neg x_2$ are literals. ANF, on the other hand, represents Boolean functions as a sum of monomials over $\mathbb{F}_2$, where addition denotes XOR and multiplication denotes AND. For instance, $x_1x_2 + x_3 + 1 = 0; x_1 + x_2 = 0$ is a ANF formula, where $x_1x_2 + x_3 + 1 = 0$ is called a clause and x_1, x_2, x_3 are called literals.

Cryptographic round function. Cipher-derived SAT formulas are typically composed of many symmetric cryptographic clause groups, where the round function serves as the fundamental building block of symmetric-key cryptographic algorithms. Each round typically consists of nonlinear operations (e.g., S-boxes), linear diffusion layers (e.g., bitwise rotations, permutations, or matrix multiplications), and key addition steps. The iterative application of the round function ensures both confusion and diffusion, which are essential for cryptographic security. For SAT-based modeling, the round function is typically encoded as a set of symmetric clauses, and we refer to the clauses corresponding to the i-th encryption round as the i-th round-function clauses.

4 Neural-guided Enumerative SAT Framework

4.1 Background & Overview

Previous methods, although highly efficient on general SAT benchmarks, often face considerable difficulties when applied to cryptographic SAT instances. As mentioned in SAT4CryptoBench [47],

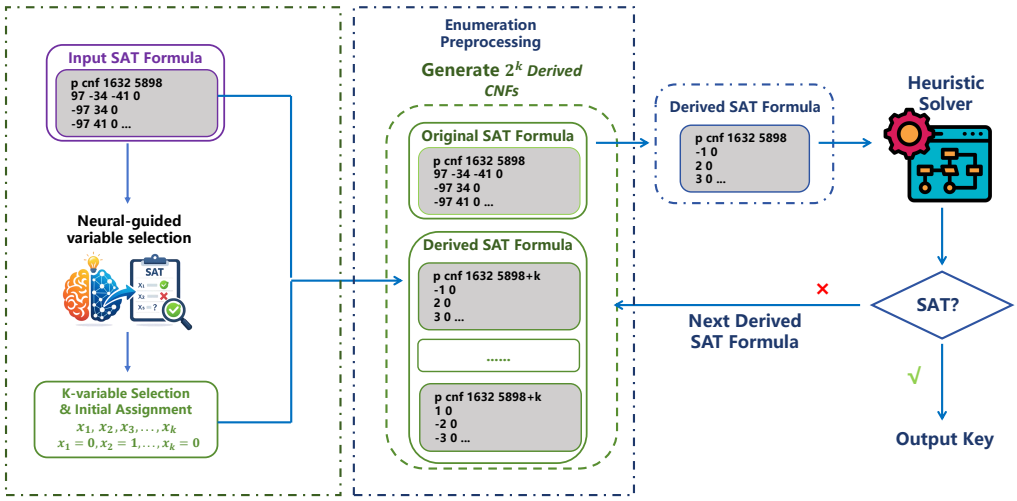


Figure 1: Overview of Neural-guided enumerative SAT framework. It first selects k critical variables and their assignments via a neural-guided variable selection module. Enumeration preprocessing then expands the original SAT instance into 2^k derived instances by enumerating these variables, and a heuristic SAT solver sequentially solves the derived instances until a satisfiable solution is found.

these difficulties mainly arise from two sources: *i*) The search space induced by round-reduced block ciphers exhibits strong symmetries, which undermine the effectiveness of standard variable activity heuristics; *ii*) Solver branching decisions tend to oscillate on variables with limited informational value, such as intermediate variables in cipher round functions, leading to exponential growth in the search process. We propose a unified neural-guided enumerative SAT framework, which introduces a *neural-guided enumeration mechanism* designed to provide more effective guidance for the solver.

Overview. The unified framework consists of three components: neural-guided variable selection, enumeration preprocessing, and a heuristic solver. As shown in Figure 1, we first select k critical variables and their initial assignments via a neural-guided variable selection module, which operates under three regimes and adapts to different settings of model capability and data prior. During enumeration preprocessing, the original instance is expanded into 2^k derived instances by enumerating these variables from their initial assignments. Then, a heuristic solver is applied sequentially to these instances until a satisfiable solution is found; otherwise, the original instance is deemed UNSAT.

Remark. In practice, the heuristic solver component can be replaced with different SAT solvers as the technology evolves. Thus, we focus on the other two components.

4.2 Enumeration Preprocessing

Instead of allowing the solver to branch on uninformative intermediate variables repeatedly, our approach focuses on fixing a subset of meaningful variables explicitly before the search begins. This idea is realized through an **enumeration preprocessing** step, where a small group of k critical variables— chosen by the neural-guided variable selection module—are explicitly assigned in advance. By expanding the original SAT instance into multiple derived instances, each of which reflects one possible assignment of these k variables, the solver is directed to explore only the reduced instances.

Formally, let F denote the original SAT formula to solve, and $V = \{x_1, x_2, \ldots, x_k\}$ be the set of selected k critical variables from neural-guided variable selection. For each assignment $\alpha = [\alpha_1, ..., \alpha_k] \in \{0,1\}^k$, we construct a derived SAT instance F_α by augmenting F with k additional unit clauses $\{C_{x_i=\alpha_i}\}$: $F_\alpha = F \wedge \bigwedge_{i=1}^{k} C_{x_i=\alpha_i}$, where each unit clause $C_{x_i=\alpha_i}$ simply fixes one variable to either 0 or 1, e.g., (x_i) or $(\neg x_i)$ in CNF and (x_i) or $(x_i + 1)$ in ANF representation.

This results in 2^k derived SAT instances F_α. Then, ordered by the Hamming distance between their corresponding assignments and the initial assignment from the neural-guided variable selection module, the derived instances are passed sequentially to the heuristic solver, effectively leveraging neural guidance. Since each instance already has the k variables determined, the solver avoids wasting search effort oscillating on these low-informative variables. In other words, equivalently,

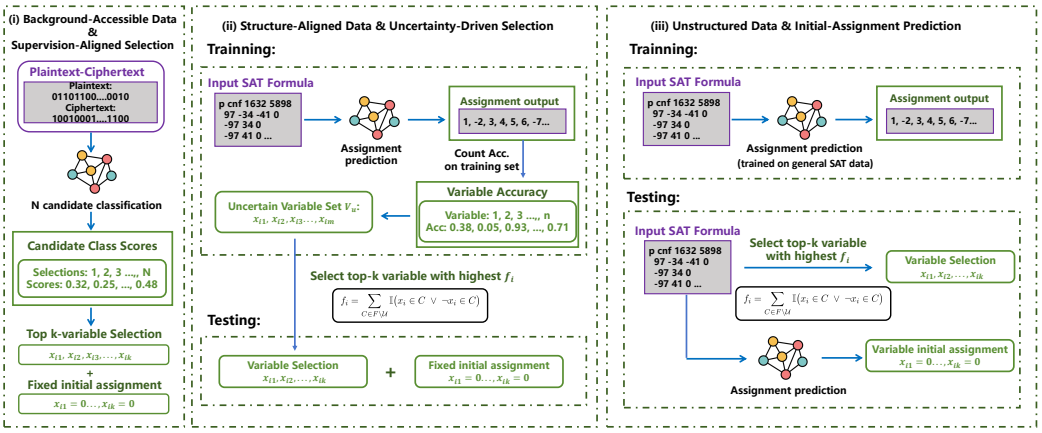


Figure 2: Overview of Neural-guided variable selection. (i) With accessible background knowledge, variable selection is formulated as an N-class classification task using plaintext–ciphertext pairs. (ii) With structure-aligned training and testing data but limited assignment prediction capability, variables are selected by first identifying uncertain variables from training-set prediction accuracy and then ranking them by occurrence frequency. (iii) With unstructured data, variables are selected according to occurrence frequency, and their initial assignments are predicted by a neural model.

the solving process on the original instance F is guided to branch on these strategically selected variables, thereby improving solving efficiency.

Importantly, this method is solver-agnostic, as the preprocessing only requires adding unit clauses and does not interfere with internal heuristics, thus making the method broadly applicable across different solvers. Besides, the parameter k introduces a trade-off between the number of derived instances and the average solving time per derived instance, which will be evaluated in Section 5.2.

4.3 Neural-guided Variable Selection

Another challenge arises from the search space induced by round-reduced block ciphers, which exhibits strong symmetries that undermine standard variable activity heuristics. In cryptographic SAT instances, symmetries appear as similar clause blocks generated by the cipher's rounds. For heuristic solvers, branching on critical variables is essential to break these symmetric structures. However, general-purpose heuristics often fail to do so consistently and instead get distracted by low-information intermediate variables. Thus, to address this issue, we propose a *neural-guided variable selection* module that selects the informative variables for enumeration.

As shown in Figure 2, in this module, the neural guidance operates under the following three regimes according to different settings of data prior and model capability.

(i) Background-Accessible Data & Supervision-Aligned Selection. In this regime, the cryptographic background of SAT instances is accessible: the underlying cipher is known, the CNF formulas share identical structure across training and inference, and their corresponding plaintext–ciphertext pairs are available. As a result, neural guidance does not need to operate at the level of assignment prediction on CNF variables; instead, strong domain priors allow us to reformulate variable selection as a classification problem over a finite set of candidates.

Formally, let n denote the number of key variables, m be the length of plaintext/ciphertext and the input to the neural network be a pair of plaintext-ciphertext bit strings (P, C), where $P, C \in \{0,1\}^m$. The network processes the concatenated bit string and outputs a probability score for candidate k-variable selections in the SAT instance. In other words, the neural network implements a mapping: $f_\theta : \{0,1\}^{2m} \rightarrow [0,1]^N$, where N is the number of candidate k-variable selections.

For training the model in this regime, we do not aim to label the solver-optimal k-variable selection because exhaustive evaluation would require prohibitive computation of all $\binom{n}{k}$ selections that grow exponentially with n. Instead, inspired by the findings in the work [38], we select the theoretically effective N k-variable candidates, where N is comparable to n. This reduces the problem to a finite multi-class classification task, making label generation tractable. Labels are generated by selecting the best-performing candidate using a heuristic solver, and the model is trained with cross-entropy

loss. The cost of generating a label for an SAT instance is equivalent to evaluating the same heuristic solver on that instance within our framework. Additional details are provided in Appendix H.

Specifically, the network f_θ consists of three parts: (i) an initial convolutional encoder that maps the concatenated plaintext–ciphertext vector (P, C) into a hidden representation, (ii) a stack of L residual blocks that extract nonlinear features from the hidden representation, and (iii) a prediction head that outputs a score for each key variable. Then, the whole selection process can be written as

$$h = \mathcal{R}_L \circ \mathcal{R}_{L-1} \circ \cdots \circ \mathcal{R}_1\big(\mathrm{IC}(P, C)\big) \tag{1}$$

$$f_\theta(P, C) = \mathrm{softmax}\big(\mathcal{H}(h)\big) \tag{2}$$

where $IC(\cdot)$ is a convolutional block and $\mathcal{H}(\cdot)$ is an MLP. $\mathcal{R}_\ell(\cdot)$ denotes the ℓ-th residual block applying a residual mapping with activation and normalization.

Here, during inference, for each SAT formula F, we choose its top k-variable selection with the highest score as the set of critical variables with fixed initial values (*e.g.*, all-zero assignment).

(ii) Structure-Aligned Data & Uncertainty-Driven Selection. In this regime, the cryptographic background is unavailable, but both training and testing use CNF instances with aligned structure. We use such data to train a model for variable assignment prediction: $g_\theta : F \rightarrow [0, 1]^n$, where F denotes the original SAT formula, n denotes the number of variables in F, and each output corresponds to the predicted probability of a variable being assigned to 1.

Importantly, this regime is designed for the practical limitations of current assignment prediction models on complex cipher-derived SAT instances. Here, g_θ exhibits limited prediction capability, with an overall accuracy of approximately 60% and highly non-uniform performance across variables. It achieves high accuracy (close to 90%) for variables that can be directly deduced, while predictions for most variables remain close to random guessing (*i.e.*, ≈ 0.5).

Based on the above observation, given the structure alignment between training and testing data, we use prediction accuracy to guide variable selection. Let p_i denote the accuracy of variable x_i on the training set. We define the set V_u of *uncertain variables* as $V_u = \{x_i | |p_i - 0.5| < \tau\}$, where τ is a predefined threshold. Furthermore, inspired by VSIDS [26], we measure structural importance via the occurrence frequency f_i of variable x_i in non-unit clauses:

$$f_i = \sum_{C \in F \setminus \mathcal{U}} \mathbb{I}\big(x_i \in C \ \vee \ \neg x_i \in C\big), \tag{3}$$

where $\mathcal{U}$ denotes the set of unit clauses in F, and $\mathbb{I}(\cdot)$ is the indicator function.

The final set of critical variables for enumeration is selected as the top-k variables in V_u with the highest f_i. This selection is fixed across all test instances along with fixed initial values of variables.

(iii) Unstructured Data & Initial-Assignment Prediction. In this regime, neither a cryptographic background nor structure-aligned data for training is available, and SAT instances may vary in size and structure. It means that there is no structural alignment between training and testing CNFs, making accuracy-based selection inapplicable. Thus, we rely entirely on a model trained on general SAT data. Similar to the (ii) regime, the model is trained to predict variable assignments: $q_\theta : F \rightarrow [0, 1]^n$.

Due to the lack of structural priors, variable selection is driven purely by clause participation. Specifically, for each SAT formula F, we calculate the frequency metric f_i defined in (ii) to rank variables, and select the top-k ones for enumeration. Differently, their initial assignments are obtained from q_θ by applying a 0.5 threshold to its predicted probabilities of variables being assigned to 1.

Notably, this regime imposes no assumptions on model prediction capacity. It means that stronger predictors naturally yield better performance, while low-accuracy models may fail to adapt effectively to the framework. This provides a robust fallback when no reliable structural information is available, enabling generalization to SAT instances in diverse backgrounds.

5 Experiments

We assess the effectiveness on both the cipher-derived cryptanalytic and general SAT datasets. The information of computation resources are given in Appendix A.

Table 1: Comparison of average runtime (s/instance) across different solvers on cryptographic datasets. NGESAT-I: Neural-guided enumerative SAT framework in the (i) regime; NGESAT-II: Neural-guided enumerative SAT framework in the (ii) regime; Random: NGESAT using random variable selection; Base: original solvers. Here, we show the 95% confidence interval.

Solver	**Kissat**				**NeuroBack-Kissat**			
	Base	**Random**	**NGESAT-I**	**NGESAT-II**	**Base**	**Random**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	778.69 ± 52.71	397.95 ± 41.25	**153.29 ± 17.46**	159.73 ± 16.88	766.58 ± 165.17	335.96 ± 42.59	177.33 ± 19.26	**166.23 ± 17.52**
Cipher-12	1017.83 ± 73.11	427.67 ± 67.27	171.51 ± 21.33	**165.38 ± 19.07**	759.26 ± 169.54	365.45 ± 47.73	185.81 ± 20.67	**164.86 ± 22.26**
Solver	**Cadical**				**CryptoMiniSat**			
	Base	**Random**	**NGESAT-I**	**NGESAT-II**	**Base**	**Random**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	1351.17 ± 241.97	603.65 ± 72.16	519.27 ± 51.39	**493.21 ± 47.04**	2684.50 ± 137.61	3543.77 ± 438.55	**533.94 ± 67.28**	597.07 ± 71.14
Cipher-12	1791.23 ± 293.97	681.11 ± 82.84	484.83 ± 48.16	**415.48 ± 50.20**	3196.25 ± 121.58	3359.47 ± 512.34	**472.91 ± 54.08**	520.99 ± 59.46
Solver	**Glucose**				**Glucose-BMM**			
	Base	**Random**	**NGESAT-I**	**NGESAT-II**	**Base**	**Random**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	1376.52 ± 200.14	567.19 ± 72.51	368.53 ± 40.72	**338.51 ± 36.08**	907.46 ± 114.28	641.15 ± 82.95	339.10 ± 37.34	**314.65 ± 33.44**
Cipher-12	825.89 ± 95.84	715.92 ± 92.58	381.74 ± 42.93	**356.68 ± 48.54**	744.71 ± 90.65	729.69 ± 95.11	373.65 ± 42.16	**350.83 ± 48.16**
Solver	**MapleSat**				**MapleSat-BMM**			
	Base	**Random**	**NGESAT-I**	**NGESAT-II**	**Base**	**Random**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	417.42 ± 52.14	793.54 ± 84.55	**319.21 ± 55.67**	362.59 ± 61.23	466.84 ± 43.25	855.61 ± 92.19	**345.27 ± 51.71**	347.58 ± 49.72
Cipher-12	429.31 ± 43.86	643.44 ± 87.16	**325.43 ± 45.51**	359.70 ± 40.57	414.21 ± 44.28	787.61 ± 93.27	360.51 ± 44.36	**355.63 ± 45.95**
Solver	**MapleCOMSPS**				**MapleCOMSPS-BMM**			
	Base	**Random**	**NGESAT-I**	**NGESAT-II**	**Base**	**Random**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	1421.65 ± 250.16	1034.98 ± 131.54	608.56 ± 66.70	**562.15 ± 59.68**	1049.04 ± 177.22	1027.69 ± 130.92	568.02 ± 62.67	**523.88 ± 55.05**
Cipher-12	925.13 ± 159.64	1167.48 ± 152.16	583.55 ± 65.42	**549.18 ± 74.71**	775.64 ± 117.10	1156.78 ± 150.58	581.66 ± 65.39	**544.65 ± 74.09**

5.1 Datasets

For cryptographic datasets, we use the Simon and Cipher benchmarks from SAT4CryptoBench [47]. Simon corresponds to SAT instances derived from the Simon block cipher, a widely studied Feistel-based construction in cryptography. In contrast, Cipher is designed to mimic ARX-based cipher decryption with unknown round functions, introducing additional difficulty due to its partially hidden structure. It evaluates solvers under ARX-dominant constraints without explicit algebraic backdoors, providing a more realistic black-box hardness. In our experiments, Simon-r-n-$2n$ denotes instances generated from the Simon cipher with r rounds, block size n, and plaintext/ciphertext length $2n$, while Cipher-k denotes instances constructed from the Cipher benchmark with k rounds.

For evaluating general SAT performance, we use a diverse and representative set of benchmark instances from recent SAT competitions. It combines instances from the past three years (2023-2025) with the benchmark [6], which offers Instances from the main track of the SAT Competition 2004 to 2022. Notably, we focus on instances that can be solved by Kissat within a time limit of 5000 seconds, ensuring both practical relevance and computational feasibility. Furthermore, we arrange the datasets (2021-2025) chronologically and divide them into three subsets—Race1, Race2, and Race3—each spanning two consecutive competition years and consisting of 150 samples. This enables us to examine performance trends across different generations of SAT instances.

5.2 Evaluation on Cryptographic Datasets

Baselines. We evaluate the effectiveness of our neural-guided enumerative SAT framework on cryptographic instances. Here, we selected Simon-12-32-64 and Cipher-12 in SAT4CryptoBench [47] as the training and testing datasets. We use 10 representative heuristic SAT solvers as baseline

Table 2: Comparison of average runtime (s/instance) between cube-and-conquer methods and the neural-guided enumerative SAT framework, with 95% confidence intervals.

Solver	Cadical				MapleSat			
	March	**AMS**	**NGESAT-I**	**NGESAT-II**	**March**	**AMS**	**NGESAT-I**	**NGESAT-II**
Simon-12-32-64	901.66 ± 135.36	977.30 ± 147.24	519.27 ± 51.39	**493.21 ± 47.04**	327.78 ± 49.76	375.19 ± 56.31	**319.21 ± 55.67**	362.59 ± 61.23
Cipher-12	949.07 ± 142.97	955.13 ± 143.09	484.83 ± 48.16	**415.48 ± 50.20**	330.21 ± 49.39	334.81 ± 50.24	**325.43 ± 45.51**	359.70 ± 40.57

Table 3: Comparison of average runtime (s/instance) on general SAT datasets. Here, we show the 95% confidence interval.

Solver	Race1	Race2	Race3
Kissat	1567.53 ± 296.10	716.27 ± 159.67	1552.19 ± 329.79
NeuroBack-Kissat	1401.93 ± 320.14	1813.62 ± 278.99	1575.59 ± 366.27
March+Kissat	1305.15 ± 276.89	1406.55 ± 214.63	2438.42 ± 370.58
AMS+Kissat	**1253.76 ± 258.09**	1343.21 ± 218.72	2331.63 ± 384.43
NGESAT-III+Kissat	1305.76 ± 291.06	**656.39 ± 191.09**	**1478.18 ± 287.65**

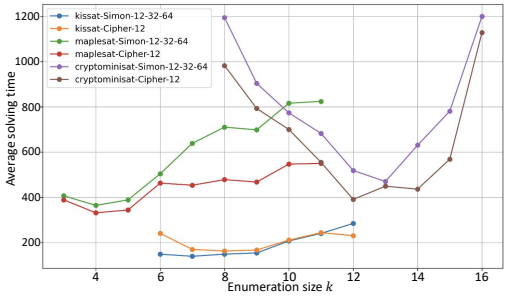


Figure 3: Enumeration Size k vs. Average solving time (s/instance) in the neural-guided enumerative SAT framework for different solvers and datasets. Average solving times are plotted over tailored k ranges for each solver.

solvers, including Kissat [12], CryptoMiniSat [36], MapleSat [27], Cadical [7], NeuroBack [40], as well as BMM-enhanced solvers [11] (including Glucose-BMM, MapleCOMSPS-BMM, and MapleSat-BMM) with their original versions. In addition, we evaluate cube-and-conquer style solvers that also rely on problem decomposition and simplification. We adopt the implementation from AlphaMapleSAT [18], testing both AMS [18] and March [15] cube strategies on Cadical and MapleSat solvers.

Settings. We compare four settings: 1) **Base**: the baseline performance of each solver on the original SAT instances; 2) **NGESAT-I**: the performance of the same solvers within our framework, where 2^k derived instances are generated, and critical variables are selected using our neural-guided variable selection module in the regime (i); 3) **NGESAT-II**: the same framework as NGESAT, but uses the neural-guided module in the regime (ii); 4) **Random**: the same framework as NGESAT, but replacing neural-guided module with random selection of k key variables. Here, we do not evaluate regime (iii) in our framework, as existing models achieve near-random assignment accuracy (around 0.5) on most variables in cryptographic datasets.

Notably, for **Base**, we record the runtime as the solving time of the original instance. For **Random**, **NGESAT-I&II**, we record the total serial solving time of the derived instances generated from the original instance, without any parallelism. For each setting, we completely solve the instances of each dataset. In addition, for regime (ii), we use NeuroBack trained on each cryptographic dataset as the limited assignment predictor g_θ. Its accuracy distribution satisfies the settings of regime (ii), making it suitable for uncertainty-driven selection. For regimes (i) and (ii), we train on 2,000 samples generated by the SAT4CryptoBench instance generator for each dataset. For regimes (i), we use the Kissat solver to generate labels. For Base, we retrain the NeuroBack on its official training dataset.

Evaluation of efficiency. For each dataset, we conduct experiments on 800 samples. Before this, we independently evaluate the choice of enumeration size by testing on 5 instances, and accordingly set the enumeration size k as follows: $k = 4$ for MapleSat and MapleSat-BMM, $k = 7$ for Kissat and NeuroBack, $k = 12$ for CryptoMiniSat, and $k = 8$ for the remaining solvers. We further discuss the trade-offs of the enumeration size k in subsequent sections.

The results in Table 1 demonstrate that our framework consistently reduces solving time compared with baseline solvers across different cryptographic SAT datasets. In particular, Kissat, Cadical, and Cryptominisat benefit significantly from our framework, achieving up to 5× speedup. Compared with the excellent prior methods in SAT4CryptoBench, our framework still attains roughly a 2× speedup.

We also compare our framework with the cube-and-conquer methods. Since cube-and-conquer methods inherently decompose problems into sub-instances for subsequent solving—a process conceptually similar to ours—we report their performance in the Base setting to ensure a fair and consistent comparison. As shown in Table 2, overall, our framework still outperforms cube-and-conquer methods, indicating that it is effective at selecting promising subspaces for search.

Besides, ablation results in Table 1 reveal that replacing neural-guided variable selection with random selection markedly degrades performance, highlighting its critical role in our framework. Moreover, we discuss the generality of our framework in Appendix G. We conduct additional experiments with the same solver, Kissat, under different optimization strategies. The results in Table 4 of Appendix G, show that our framework consistently provides substantial speedups across optimization methods.

Evaluation of enumeration size k's Trade-off. In addition, we conduct experiments on 100 samples to analyze the effect of the enumeration size k. A larger k generates more derived instances (2^k), increasing computational cost but enhancing simplification and solver efficiency, while a smaller k reduces overhead at the expense of weaker symmetry breaking. Here, we use three representative SAT solvers, including Kissat, CryptoMiniSat, and MapleSat, as baselines for comparison. These solvers are widely used for large and complex SAT instances. We vary k and plot solver runtimes on different datasets within our framework. As shown in Figure 3, we find that across solvers and datasets, runtime first decreases and then increases as k grows, indicating the existence of a practical balance between overhead and effectiveness. We further discuss the choice of k in Section 5.4.

5.3 Evaluation on General Datasets

Baselines & Settings. We evaluate the generalization ability of our framework on general SAT datasets, using Race1, Race2, and Race3 constructed from the SAT Competition. For comparison, we adopt the representative solver Kissat [12] and its variant NeuroBack [40] as baselines.

Since no training data aligned with the test distribution is available, we consider five settings: the baseline performance of 1) Kissat and 2) NeuroBack; Kissat combined with 3) AMS and 4) March-based cubing strategies; and 5) the performance of Kissat within our framework under regime (iii), where NeuroBack is used to predict variable assignments. In this setting, due to the absence of a pretrained model, we train NeuroBack on its official training dataset.

Evaluation of efficiency. Before evaluation, due to the absence of any prior information on the dataset, we directly follow the optimal setting of the evaluation of enumeration size k's trade-off to set $k = 7$. The results in Table 3 indicate that our framework exhibits a certain degree of generalization to general SAT problems. Even when using the same assignment prediction model, our method consistently achieves better performance on general SAT datasets. Moreover, compared with cube-and-conquer strategies, our framework delivers comparable performance, suggesting that it can guide the search toward promising regions of the solution space with similar effectiveness.

5.4 Discussion about the choice of enumeration size k

In the neural-guided enumerative SAT framework, k is treated as a structural hyperparameter. Specifically, the total runtime can be written as $T_{\text{total}}(k) = 2^k \cdot T_{\text{solve}}(F_\alpha)$, where 2^k denotes the number of enumerated sub-instances, $T_{\text{solve}}(F_\alpha)$ is the average solving time per sub-instance. As k increases, stronger symmetry breaking and variable enumerating typically reduce $T_{\text{solve}}(F_\alpha)$, while the enumeration cost grows exponentially with 2^k. Therefore, **the value of k does not grow with the hardness or runtime of the original instance, but instead remains within a relatively small and stable range** (typically $k \in [4, 12]$). Then, we adopt a lightweight validation-based strategy for parameter selection. Specifically, we evaluate the average runtime on a small independent validation set over a limited search range (e.g., $k \in [4, 12]$) and select the value achieving the best trade-off.

6 Discussion of Neural Guidance Regimes

In this section, we discuss the rationale behind the three neural guidance regimes introduced in Section 4.3, drawing connections between cryptographic structure, neural model behavior, and established SAT solving heuristics. This analysis offers a unified perspective on why the enumeration framework described above is effective across these problem settings.

Regime (i): Supervised Selection with Data Priors. The candidate variable selections in Regime (i) are constrained by cryptographic structure rather than chosen arbitrarily. As shown by Trimoska et al. [38], for cryptographic SAT instances, prioritizing the enumeration of variables whose assignments can reduce high-order terms in the cipher's round function to lower-order terms can significantly decrease solving time. Due to the inherent cyclic symmetry of the round function, these variables form a cyclic sequence whose length equals the number of key variables, and selecting a consecutive segment from this cycle constitutes a structurally grounded and near-optimal strategy. Consequently, the candidate space is reduced to a finite set of n possible selections, and our neural-guided variable selection is formulated as a classification problem over these structurally constrained candidates rather than an unconstrained search over all $\binom{n}{k}$ subsets.

Regime (ii): Uncertainty-Driven Selection. The selection of variables with near-random prediction accuracy (≈ 0.5) in Regime (ii) is motivated by the interplay between neural model behavior and aligned structure. Interpretability analyses of NeuroSAT [31] and NeuroGIFT [37] reveal that the learning process of neural SAT models typically proceeds in stages: assignments for variables that can be deduced through shallow reasoning are learned first, while variables requiring deeper inference are learned later or remain unresolved. Notably, as observed in NeuroGIFT [37], variables associated with cryptographic nonlinear operations (e.g., the S-box or modular multiplication terms in round functions) exhibit minimal changes in their learned representations throughout training, indicating that these variables are inherently more difficult for neural models to predict and thus consistently yield near-random prediction accuracy. This observation aligns with the structural analysis in [38]: the nonlinear aligned structures that neural models struggle to predict correspond precisely to the high-order terms whose enumeration is most beneficial for reducing solver runtime. By further incorporating occurrence frequency f_i in non-unit clauses—a classical heuristic measure inspired by VSIDS [26]—Regime (ii) combines neural uncertainty with structural importance, yielding a variable selection strategy that is supported by both neural model behavior and aligned structure analysis. More detailed analyses are provided in Appendix I.

Regime (iii): Robust Fallback for General SAT. Regime (iii) addresses settings where neither cryptographic priors nor structural alignment between training and testing data is available. In such cases, directly identifying the optimal variable subset is generally infeasible without domain-specific knowledge. Our approach leverages the representation capability of a neural model trained on general SAT data, combined with the heuristic frequency metric f_i, to identify effective suboptimal variable selections. This follows the well-established principle of heuristic SAT solver design [26], where practical performance is continuously improved by discovering and exploiting informative search patterns rather than requiring exact optimal decisions. In this sense, Regime (iii) can be viewed as a natural extension of prior neural-guided SAT approaches such as NeuroBack [40] within our enumeration framework, providing a robust fallback that maintains applicability and reasonable performance across diverse problem settings. Taken together, the three regimes demonstrate that the effectiveness of our framework stems not from a single universal heuristic, but from aligning the variable selection strategy with the structural characteristics of the problem at hand.

7 Conclusion

In this paper, We propose a unified neural-guided enumerative SAT framework for cryptographic key recovery that combines learning-based variable selection with structured enumeration. By focusing the search on a small set of critical variables, the method reduces combinatorial complexity with minimal neural overhead. We further unify different neural guidance regimes to improve robustness across settings. Experiments show consistent speedups and improved performance on cryptographic and general SAT problems, highlighting the effectiveness and generalization.

Limitation & Future work. Although the proposed framework is effective, its performance still partially depends on the behavior and robustness of the underlying heuristic SAT solver, which may introduce variability across different instance distributions. In addition, the current design separates neural variable selection and symbolic enumeration, which may limit the degree of end-to-end optimization. In future work, we aim to integrate the enumeration paradigm into an end-to-end differentiable discriminator, enabling joint optimization of variable selection and key recovery for improved accuracy and stability. As a preliminary illustration of this direction, we provide an example on the Simon cipher in Appendix D. More limitations can be found in Appendix B.

References

[1] Albert Atserias, Johannes Klaus Fichte, and Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution. *Journal of Artificial Intelligence Research*, 40:353–373, 2011.

[2] Zhenzhen Bao, Jian Guo, Meicheng Liu, Li Ma, and Yi Tu. Enhancing differential-neural cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 318–347. Springer, 2022.

[3] Zhenzhen Bao, Jinyu Lu, Yiran Yao, and Liu Zhang. More insight on deep learning-aided cryptanalysis. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 436–467. Springer, 2023.

[4] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow—resolution made simple. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 517–526, 1999.

[5] Adrien Benamira, David Gerault, Thomas Peyrin, and Quan Quan Tan. A deeper look at machine learning-based cryptanalysis. In *Advances in Cryptology–EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part I 40*, pages 805–835. Springer, 2021.

[6] A Biere, N Froleyks, and W Wang. Sampled and normalized satisfiable instances from the main track of the sat competition 2004 to 2022, 2023.

[7] Armin Biere, Tobias Faller, Katalin Fazekas, Mathias Fleury, Nils Froleyks, and Florian Pollitt. Cadical 2.0. In *International Conference on Computer Aided Verification*, pages 133–152. Springer, 2024.

[8] Benedikt Bünz and Matthew Lamm. Graph neural networks and boolean satisfiability. *arXiv preprint arXiv:1702.03592*, 2017.

[9] Chris Cameron, Rex Chen, Jason Hartford, and Kevin Leyton-Brown. Predicting propositional satisfiability via end-to-end learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3324–3331, 2020.

[10] Kamil Dreczkowski, Antoine Grosnit, and Haitham Bou Ammar. Framework and benchmarks for combinatorial and mixed-variable bayesian optimization. *Advances in Neural Information Processing Systems*, 36:69464–69489, 2023.

[11] Haonan Duan, Saeed Nejati, George Trimponias, Pascal Poupart, and Vijay Ganesh. Online bayesian moment matching based sat solver heuristics. In *International Conference on Machine Learning*, pages 2710–2719. PMLR, 2020.

[12] Armin Biere Katalin Fazekas Mathias Fleury and Maximilian Heisinger. Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020. *SAT COMPETITION*, 50:2020, 2020.

[13] Aron Gohr. Improving attacks on round-reduced speck32/64 using deep learning. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part II 39*, pages 150–179. Springer, 2019.

[14] Jesse Michael Han. Enhancing sat solvers with glue variable predictions. *arXiv preprint arXiv:2007.02559*, 2020.

[15] Marijn J. H. Heule, Oliver Kullmann, Siert Wieringa, and Armin Biere. Cube and conquer: Guiding cdcl sat solvers by lookaheads. In *Haifa Verification Conference*, volume 7261 of *Lecture Notes in Computer Science*, pages 50–65, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[16] Zezhou Hou, Jiongjiong Ren, and Shaozhen Chen. Cryptanalysis of round-reduced simon32 based on deep learning. *Cryptology ePrint Archive*, 2021.

[17] Sebastian Jaszczur, Michał Łuszczyk, and Henryk Michalewski. Neural heuristics for sat solving. *arXiv preprint arXiv:2005.13406*, 2020.

[18] Piyush Jha, Zhengyu Li, Zhengyang Lu, Curtis Bright, and Vijay Ganesh. Alphamaplesat: An mcts-based cube-and-conquer sat solver for hard combinatorial problems. *arXiv preprint arXiv:2401.13770*, 2024.

[19] Vitaly Kurin, Saad Godil, Shimon Whiteson, and Bryan Catanzaro. Can q-learning with graph networks learn a generalizable branching heuristic for a sat solver? *Advances in Neural Information Processing Systems*, 33:9608–9621, 2020.

[20] Yang Li, Yuqi Jia, Wanqian Luo, Hui-Ling Zhen, Xijun Li, Mingxuan Yuan, and Junchi Yan. Kissat adaptive restart, kissat cfexp: Adaptive restart policy and variable scoring improvement. *Proceedings of SAT Competition*, 2022.

[21] Zhaoyu Li, Jinpei Guo, and Xujie Si. G4satbench: Benchmarking and advancing sat solving with graph neural networks, 2024.

[22] Zhaoyu Li and Xujie Si. Nsnet: A general neural probabilistic framework for satisfiability problems. *Advances in Neural Information Processing Systems*, 35:25573–25585, 2022.

[23] Jia Hui Liang, Vijay Ganesh, Ed Zulkoski, Atulan Zaman, and Krzysztof Czarnecki. Understanding vsids branching heuristics in conflict-driven clause-learning sat solvers. In *Haifa Verification Conference*, pages 225–241. Springer, 2015.

[24] Jia Hui Liang, Chanseok Oh, Minu Mathew, Ciza Thomas, Chunxiao Li, and Vijay Ganesh. Machine learning-based restart policy for cdcl sat solvers. In *Theory and Applications of Satisfiability Testing–SAT 2018: 21st International Conference, SAT 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 9–12, 2018, Proceedings 21*, pages 94–110. Springer, 2018.

[25] David Mojžíšek, Jan Hla, Ziwei Li, Ziyu Zhou, and Mikoláš Janota. Neural approaches to sat solving: Design choices and interpretability. *International Journal of Approximate Reasoning*, page 109609, 2025.

[26] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient sat solver. In *Proceedings of the 38th annual Design Automation Conference*, pages 530–535. ACM, 2001.

[27] Saeed Nejati, Md Solimul Chowdhury, and Vijay Ganesh. Maplessv sat solver for sat competition 2021. *SAT COMPETITION 2021*, page 35, 2021.

[28] Saeed Nejati and Vijay Ganesh. Cdcl (crypto) sat solvers for cryptanalysis. *arXiv preprint arXiv:2005.13415*, 2020.

[29] Saeed Nejati, Ludovic Le Frioux, and Vijay Ganesh. A machine learning based splitting heuristic for divide-and-conquer solvers. In *Principles and Practice of Constraint Programming: 26th International Conference, CP 2020, Louvain-la-Neuve, Belgium, September 7–11, 2020, Proceedings 26*, pages 899–916. Springer, 2020.

[30] Knot Pipatsrisawat and Adnan Darwiche. On the power of clause-learning sat solvers as resolution engines. *Artificial intelligence*, 175(2):512–525, 2011.

[31] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a sat solver from single-bit supervision. In *International Conference on Learning Representations*, 2019.

[32] Byoungjin Seok and Changhoon Lee. A novel approach to construct a good dataset for differential-neural cryptanalysis. *IEEE Transactions on Dependable and Secure Computing*, 22(1):246–262, 2024.

[33] Zhengyuan Shi, Min Li, Yi Liu, Sadaf Khan, Junhua Huang, Hui-Ling Zhen, Mingxuan Yuan, and Qiang Xu. Satformer: Transformer-based unsat core learning. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–4. IEEE, 2023.

[34] Ajeet Singh, Kaushik Bhargav Sivangi, and Appala Naidu Tentu. Machine learning and cryptanalysis: An in-depth exploration of current practices and future potential. *Journal of Computing Theories and Applications*, 2(1):27–42, 2024.

[35] Jaewoo So. Deep learning-based cryptanalysis of lightweight block ciphers. *Security and Communication Networks*, 2020(1):3701067, 2020.

[36] Mate Soos. The cryptominisat 5 set of solvers at sat competition 2016. *Proceedings of SAT Competition*, page 28, 2016.

[37] Ling Sun, David Gerault, Adrien Benamira, and Thomas Peyrin. Neurogift: Using a machine learning based sat solver for cryptanalysis. In *International Symposium on Cyber Security Cryptography and Machine Learning*, pages 62–84. Springer, 2020.

[38] Monika Trimoska, Sorina Ionica, and Gilles Dequen. *Parity (XOR) Reasoning for the Index Calculus Attack*, page 774–790. Springer International Publishing, 2020.

[39] Pashootan Vaezipoor, Gil Lederman, Yuhuai Wu, Roger Grosse, and Fahiem Bacchus. Learning clause deletion heuristics with reinforcement learning. In *5th Conference on Artificial Intelligence and Theorem Proving*, page 80, 2020.

[40] Wenxi Wang, Yang Hu, Mohit Tiwari, Sarfraz Khurshid, Kenneth McMillan, and Risto Miikkulainen. Neuroback: Improving cdcl sat solving using graph neural networks. *arXiv preprint arXiv:2110.14053*, 2021.

[41] Xiaoxing Wang, Zhirui Lian, Jiale Lin, Chao Xue, and Junchi Yan. Diy your easynas for vision: Convolution operation merging, map channel reducing, and search space to supernet conversion tooling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(11):13974–13990, 2023.

[42] Ryan Williams, Carla P Gomes, and Bart Selman. Backdoors to typical case complexity. In *IJCAI*, volume 3, pages 1173–1178, 2003.

[43] Zehan Wu, Kexin Qiao, Zhaoyang Wang , Junjie Cheng , and Liehuang Zhu . Mixture differential cryptanalysis on round-reduced simon32/64 using machine learning. *Mathematics*, 12(9), 2024.

[44] Zhiyuan Yan, Min Li, Zhengyuan Shi, Wenjie Zhang, Yingcong Chen, and Hongce Zhang. Addressing variable dependency in gnn-based sat solving. *arXiv preprint arXiv:2304.08738*, 2023.

[45] Ziwei Zhang and Yang Zhang. Elimination mechanism of glue variables for solving sat problems in linguistics. In *The Asian Conference on Language*, pages 147–167, 2021.

[46] Xinhao Zheng, Yang Li, Cunxin Fan, Huaijin Wu, Xinhao Song, and Junchi Yan. Learning plaintext-ciphertext cryptographic problems via anf-based sat instance representation. In *Advances in Neural Information Processing Systems*, pages 41649–41669, 2024.

[47] Xinhao Zheng, Xinhao Song, Bolin Qiu, Zhongteng Gui, and Junchi Yan. Bridging crypto with ml-based solvers: the sat formulation and benchmarks. In *Advances in Neural Information Processing Systems*, 2025.

A Computational Resources

All experiments were conducted on a dedicated high-performance computing cluster equipped with NVIDIA A100 GPUs. Training and inference for all neural components were performed entirely on these GPUs using the PyTorch framework, with CUDA and cuDNN enabled for efficient acceleration. CPU resources were used only for data loading and SAT-solving routines, while GPU devices were exclusively allocated to neural network training to avoid inter-process contention. The neural network itself is lightweight and computationally efficient, imposing only minimal demands on GPU resources and allowing fast and stable training.

B Limitations

While the proposed framework demonstrates strong effectiveness and generalization across both cryptographic and general SAT benchmarks, several limitations remain. First, the overall performance still depends on the behavior of the underlying heuristic SAT solver, whose search dynamics may vary across different instance distributions and solver configurations. Second, the current framework adopts a modular design that separates neural variable selection from symbolic enumeration and SAT solving. Although this separation improves flexibility and efficiency, it may limit deeper end-to-end coordination between learned guidance and symbolic reasoning.

C Impact statement

This paper advances machine learning by combining neural guidance with SAT-based cryptanalysis, using neural models to guide enumeration and prune incorrect assignments, improving efficiency on Simon instances. Our work contributes to neural-assisted symbolic reasoning and cryptographic evaluation. While cryptanalysis may have security implications, this research is intended for scientific purposes, and we do not foresee ethical concerns beyond those commonly associated with the fields.

D Application: Enumeration for neural distinction in SAT solving

D.1 Background & Overview

Existing neural-discriminator-based methods for key recovery still face important challenges. Differential distinguishers require costly enumeration and large training datasets, while lacking generality for single plaintext–ciphertext scenarios. Likewise, neural SAT classifiers tend to focus on satisfiability prediction to distinguish correct from incorrect one-bit key assignments for key solving, but their bit-level distinction accuracy remains modest for reliable key recovery. Faced with these challenges, we propose an example application of enumeration for neural distinction in SAT solving on the Simon dataset, which enumerates k key variables across different assignments and selection based on cipher-induced structural clauses and introduces a *neural-guided distinction mechanism* to enable full key recovery via unit propagation.

Overview. The whole framework consists of two parts: candidate derived instance generation and neural-guided distinction solving. As shown in Figure 4, we first determine the length of the enumerated assignment as the number of key variables required to fully simplify the relevant shallow round-function clauses. Then, we enumerate 2^k assignments. For each assignment, based on the assignment length analysis, we select key variables that fully simplify the relative round-function clauses, forming the corresponding variable selection. Each assignment is substituted into the original instance, and this yields 2^k derived instances. Then we use a neural network to distinguish the correct instance and recover all the variables by propagating known values and extracting assignments via unit clauses (e.g., $x_1 = 0$).

Remark. Since ANF provides a more concise representation than CNF, we adopt ANF as the SAT representation, and SAT instances in this section are assumed to be in ANF.

D.2 Candidate derived instance generation

The limited recovery accuracy of existing neural SAT approaches often stems from the minimal differences between derived instances produced by single-bit key assignments. For example, in the ANF representation, the distinction between two corresponding derived instances only lies in the

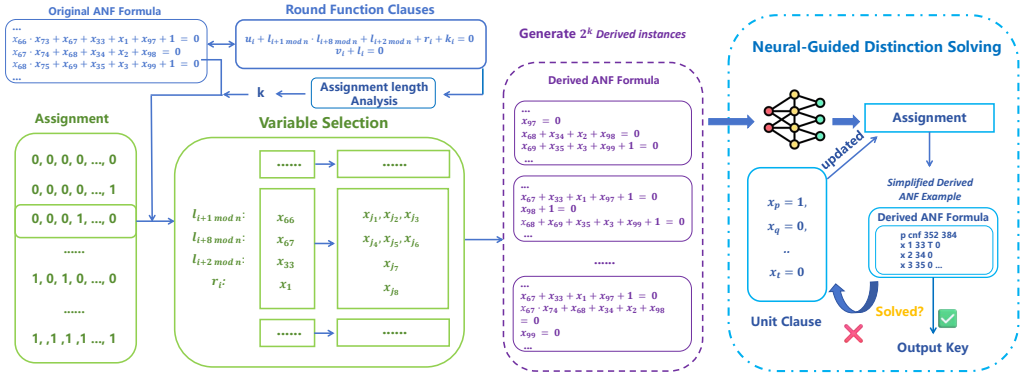


Figure 4: Overview of the application of enumeration for neural distinction in SAT solving on the Simon dataset. The framework first determines the assignment length k by analyzing cipher-induced symmetric round function clauses. Based on the assignment length analysis, for each candidate assignment, the corresponding variable selection is derived from the round-function clauses, yielding 2^k candidate instances. A neural distinction model then separates correct from incorrect instances, and iterative unit propagation on the correct instance recovers the full secret key.

constant term of a single clause, while the majority of clauses, which are structurally symmetric due to the cipher's round functions, remain identical. Therefore, we focus on breaking the symmetries of round-function clauses to generate candidate derived instances with greater distinction.

Here, we take the round-function clauses of the Simon dataset from SAT4CryptoBench [47] as an example, where each round uses the same seed key. Let $L = \overline{l_0, l_1, \ldots, l_{n-1}}$ denote the left (high-order) input of the round function, $R = \overline{r_0, r_1, \ldots, r_{n-1}}$ the right (low-order) input, $K = \overline{k_0, k_1, \ldots, u_{n-1}}$ the round key, $U = \overline{u_0, u_1, \ldots, u_{n-1}}$ the left (high-order) output, and $V = \overline{v_0, v_1, \ldots, v_{n-1}}$ the right (low-order) output, where n is the block size and $l_i, r_i, k_i, u_i, v_i \in \{0, 1\}$.

Assignment length Analysis. We first analyse the symmetric round-function clauses to determine the assignment length k. The round function of Simon is shown as follows:

$$\begin{aligned} U &= ((L \lll 1) \cdot (L \lll 8)) \\ &\quad \oplus (L \lll 2) \oplus R \oplus K \\ V &= L \end{aligned} \tag{4}$$

The corresponding round-function clauses in the ANF are given as follows, for $i = 0, 1, \ldots, n-1$:

$$\begin{aligned} u_i + l_{i+1 \mod n} \cdot l_{i+8 \mod n} \\ + l_{i+2 \mod n} + r_i + k_i = 0 \\ v_i + l_i = 0 \end{aligned} \tag{5}$$

Each round introduces n structurally symmetric clauses. Since clauses in the middle rounds typically involve a larger number of key variables, we focus on the first three and the last three rounds. Guided by the following reasoning, we set $k = 8$ as the assignment length.

Analysis

In the first round, V corresponds to the plaintext and is constant, while each bit in U depends only on a single key bit. In the second round, every bit in L is influenced by one key bit, while R remains constant. Hence, knowing the values of just three key bits—$l_{i+1 \bmod n}$, $l_{i+8 \bmod n}$, and k_i—is sufficient to determine u_i. In the third round, the dependency becomes richer: each bit in L involves 3 key variables, and R is tied to one key bit. To fully simplify the i-th clause in this round, we must determine $l_{i+2 \bmod n}$, r_i, and k_i, and choose an assignment that forces either $l_{i+1 \bmod n}$ or $l_{i+8 \bmod n}$ to 0.
For the last three rounds, the symmetry of Simon yields similar results. Specifically, once the i-th clause in the third round is fully simplified, the corresponding clause in the third-to-last round is also fully simplified. This shows that at least 3+3+1+1=8 variables are required.

Generation Process. Based on the assignment length analysis, as shown in Figure 4, we generate the derived instances in three steps. First, we check the special case where $l_i = 1$ for all $i \in 1, 2, \ldots, n$ in the third-round clauses, which only requires verifying simple linear Boolean equations from the first two rounds' clauses. If this condition holds, the solution can be obtained directly. Second, we generate all 2^k possible k-bit assignments. Third, for each specific assignment, we substitute the k-bit assignment into the i-th ($i = 1, \ldots, n$) third round-function clauses to compute $l_{i+1 \bmod n}, l_{i+2 \bmod n}$, r_i, and k_i, and evaluate the resulting candidates. The instance with the greatest simplification is chosen as the derived instance for that assignment.

Remark. Notably, in the first step, we handle a special case to ensure that at least one of these derived instances is satisfiable, and this is proved in Appendix F. Moreover, in practice, since the type of round function clauses is fixed, we only need a finite table to generate derived instances, making the overhead negligible.

D.3 Neural-guided distinction solving

After generating the 2^k derived SAT instances, we use a neural network to distinguish the correct instance. Formally, let F be the original SAT instance in ANF and $\mathcal{A}$ be the assignment set of the derived SAT instances. For each assignment $a = [a_1, ..., a_k] \in \{0, 1\}^k$ with its related subset of key variables $V_a = \{x_{i_1}, ..., x_{i_k}\}$, the derived instance is constructed as: $F_a = F \ \wedge \ \bigwedge_{j=1}^{k} C_{x_{i_j}=a_j}$.

To distinguish these instances, we train a GNN-based neural network $\mathcal{N}_\theta$ to map an ANF instance to a confidence score $s_a \in [0, 1]$ indicating whether a is likely to correspond to a correct partial key assignment: $s_a = \mathcal{N}_\theta(F_a), \quad a \in \mathcal{A}$. Specifically, the network $\mathcal{N}_\theta$ converts the SAT instance into an ANF graph proposed in the prior work [46], iteratively refines the embeddings of literals and clauses, and outputs the classification logits through a linear classifier, as follows:

$$\begin{aligned} \textbf{Iteration:} \quad & h_l^{(t)} = \text{AGG}_l\left(\left\{h_c^{(t-1)} | c \in \mathcal{N}(l)\right\}, h_l^{(t-1)}\right) \\ & h_{c,\text{pos}}^{(t)}, h_{c,\text{neg}}^{(t)} = \text{AGG}_c\left(\left\{h_l^{(t-1)} | l \in \mathcal{N}(c)\right\}, h_c^{(t-1)}\right) \\ \textbf{Linear classifier:} \quad & s_a = C_{\text{vote}}(h_{c,\text{pos}}, h_{c,\text{neg}}) \end{aligned} \tag{6}$$

where $\mathcal{N}()$ denotes the set of the neighbor nodes. $C_{\text{vote}}(\cdot)$ denotes an MLP, and $\text{AGG}(\cdot)$ represents the aggregation function. $h_{c,\text{pos}}^{(t)}, h_{c,\text{neg}}^{(t)}$ represent the embeddings of the t-th iteration for the positive clauses and the relevant negative clauses, respectively. $h_{c,\text{pos}}, h_{c,\text{neg}}$ represent the embeddings of the final iteration. Then, during solving, the network evaluates all derived instances $\{F_a\}$, and the assignment with the highest confidence score is selected: $a^* = \arg\max_{a \in \mathcal{A}} s_a$.

Unit propagation. Given the selected k-bit assignment $a^* = [a_1^*, \ldots, a_k^*]$, we recover all variables in the SAT instance through iterative unit propagation. Starting from the solution set $S = \{x_{i_1} = a_1^*, \ldots, x_{i_k} = a_k^*\}$, we substitute known values into clauses (e.g., $x_{i_1} + x_3 = 0$ becomes $x_3 + a_1^* = 0$) and extract new assignments from the resulting unit clauses, such as $x_3 + a_1^* = 0$, which contains only a single variable, thereby expanding S. This process iterates until all variables are determined or a conflict occurs. When no unit clause is available, we assign 0 to the first variable in the shortest clause to continue. Although this heuristic may slightly affect recovery accuracy, the method still achieves a high full-recovery success rate.

In practice, we use the aggregation functions of the ANF-based network in the prior work [46], and the detailed structure is shown in Appendix E. Besides, we use the classification label to train this network and use the logits corresponding to satisfiable as the confidence scores.

E ANF-based Model Architecture

This section provides the complete architectural details of the network we use in Appendix D, which is an ANF-based network in SAT4CryptoBench [47], including the message-passing mechanism and the attention-based prediction head.

Message-Passing Network We randomly initialize embeddings for vanilla literals and paired complementary clause nodes. The message-passing process consists of T iterations, during which node embeddings are iteratively updated by aggregating information from their graph neighbors.

In each iteration, clause embeddings are first updated. A clause node receives messages from connected monomial nodes, which themselves aggregate messages from literal nodes, as well as from their complementary clause node. The update equations are as follows:

$$\begin{aligned} L_{l2c}^{(t)} &= L_{l2l}(M_{l2l}^T L^{(t)}) \\ [C_{m,\text{pos}}^{(t)}, C_{m,\text{neg}}^{(t)}] &= M_{l2c}^T L_{\text{msg}}(L_{l2c}^{(t)}) \\ (C_{\text{pos}}^{(t+1)}, C_{h,\text{pos}}^{(t+1)}) &\leftarrow C_{u,\text{pos}}([C_{h,\text{pos}}^{(t)}, C_{\text{neg}}^{(t)}, C_{m,\text{pos}}^{(t)}]) \\ (C_{\text{neg}}^{(t+1)}, C_{h,\text{neg}}^{(t+1)}) &\leftarrow C_{u,\text{neg}}([C_{h,\text{neg}}^{(t)}, C_{\text{pos}}^{(t)}, C_{m,\text{neg}}^{(t)}]) \end{aligned} \tag{7}$$

Here, M_{l2l} and M_{l2c} denote sparse adjacency matrices connecting literals to monomials and literals to clauses, respectively. The functions L_{l2l} and L_{msg} are multilayer perceptrons (MLPs) applied to literal messages. Clause updates are performed via layer-normalized LSTMs, $C_{u,\text{pos}}$ and $C_{u,\text{neg}}$.

Next, literal embeddings are updated by receiving messages from clauses, via intermediate monomial aggregation:

$$L_{c2l}^{(t)} = M_{l2c} C_{\text{msg}}([C_{\text{pos}}^{(t)}, C_{\text{neg}}^{(t)}]) \tag{8}$$

$$L_m^{(t)} = M_{l2l} L_{l2m}(L_{c2l}^{(t)}) \tag{9}$$

$$(L^{(t+1)}, L_h^{(t+1)}) \leftarrow L_u([L_h^{(t)}, L_m^{(t)}]) \tag{10}$$

where C_{msg} and L_{l2m} are MLPs, and L_u is a layer-normalized LSTM that updates the literal state.

Feedforward Prediction Head with Attention After T rounds of message passing, the refined clause embeddings are used to predict the satisfiability of the formula. Unlike [46], which aggregates clause embeddings by simple summation, we incorporate an attention mechanism inspired by Graph Attention Networks (GAT). Specifically, attention weights are computed across all positive and negative clause nodes to obtain a weighted aggregation:

$$\alpha_i = \frac{\exp(\text{LeakyReLU}(a^\top [W_c c_i \,\|\, W_c \bar{c}_i]))}{\sum_j \exp(\text{LeakyReLU}(a^\top [W_c c_j \,\|\, W_c \bar{c}_j]))} \tag{11}$$

$$s = \sigma\left(\sum_i \alpha_i \cdot (W_c c_i + W_c \bar{c}_i)\right) \tag{12}$$

where c_i and $\bar{c}_i$ denote the i-th positive and negative clause embeddings, respectively, W_c is a learnable projection matrix, a is the attention vector, and $\|\cdot\|$ denotes vector concatenation. The final satisfiability score $s \in [0, 1]$ is obtained by applying the sigmoid activation $\sigma(\cdot)$.

The model is trained with binary cross-entropy loss between the predicted score s and the ground truth label $y \in \{0, 1\}$.

F Proof of the completeness for Candidate derived instance generation

Completeness of Generation process: The generation process is shown as follows:

1. Check the special case where $l_i = 1$ for all $i \in 1, 2, \ldots, n$ in the third-round clauses, which only requires verifying simple linear Boolean equations from the first two rounds clauses. If this condition holds, the solution can be obtained directly.
2. Generate all 2^k possible k-bit assignments.
3. For each specific assignment, substitute the k-bit assignment into the i-th ($i = 1, \ldots, n$) third-round clauses to compute $l_{i+1 \bmod n}$, $l_{i+2 \bmod n}$, r_i, and k_i, and evaluate the resulting candidates. The instance with the greatest simplification is chosen as the derived instance for that assignment.

where the corresponding round-function clauses in the ANF are given as follows, for $i = 0, 1, \ldots, m-1$: $u_i + l_{i+1 \bmod n} \cdot l_{i+8 \bmod n} + l_{i+2 \bmod n} + r_i + k_i = 0,\ v_i + l_i = 0$.

Proof. The completeness of the generation process implies that the correct derived instance is always included among the generated 2^k instances. Therefore, it suffices to prove that at least one satisfiable instance exists within the 2^k derived instances.

Based on the completeness of the generation process, we consider the following two cases:

1. **All third-round key variables satisfy $l_i = 1$ for $i \in \{1, 2, \ldots, n\}$.**
   In this scenario, the correct key is obtained directly in the first step.
2. **Not all third-round key variables satisfy $l_i = 1$ for $i \in \{1, 2, \ldots, n\}$.**
   In this case, there exists a k-bit assignment such that $l_{p+1 \bmod n} = 0$ for some $p \in \{1, 2, \ldots, n\}$. For the corresponding round-function clause containing $l_{p+1 \bmod n}$, substituting this k-bit assignment ensures that $l_{p+1 \bmod n}$, $l_{p+2 \bmod n}$, r_p, and k_p are all known. Moreover, since $l_{p+1 \bmod n} = 0$, we have

   $$l_{p+1 \bmod n} \cdot l_{p+8 \bmod n} = 0,$$

   So this clause is fully simplified under this assignment. (In practice, even when multiple clauses achieve the same level of simplification, a few steps of unit propagation are sufficient to identify satisfiable instances that the network can reliably distinguish.) Therefore, the instance corresponding to the true key variables must appear among the generated instances.

Combined with the above two cases, we have proved the completeness of the generation process. □

G Additional Experiments on the generality of Neural-guided Enumerative SAT Framework

We further test Kissat under different optimization strategies, including the adaptive restart policy [20] and the hyperparameter optimization [10]. As shown in Table 4 (with 95% confidence interval), it consistently delivers substantial speedups across these strategies. These results highlight the robustness and generality of our framework: regardless of the specific optimization strategies embedded in solvers, our framework consistently provides notable performance gains, further underscoring the essential role in accelerating SAT solving on cryptographic instances.

Table 4: Comparison of average runtime (s/instance) for solvers under different optimization strategies across frameworks. **Base**: the baseline performance of each solver on the original SAT instances; **NGESAT-I**: the performance of the same solvers within the neural-guided enumerative SAT framework (NGESAT), where 2^k derived instances are generated, and critical variables are selected using our neural-guided variable selection module in the regime (i); **NGESAT-II**: the same framework as NGESAT, but uses the neural-guided module in the regime (ii)

Solver	Methods	Simon 12-32-64	Cipher-12
Kissat	Base	778.69 ± 52.71	1017.83 ± 73.11
	Random	397.95 ± 41.25	427.67 ± 67.27
	NGESAT-I	**153.29 ± 17.46**	**171.51 ± 21.33**
	NGESAT-II	**159.73 ± 16.88**	**165.38 ± 19.07**
Kissat-restart	Base	349.29 ± 69.35	367.70 ± 61.31
	Random	455.08 ± 57.43	471.26 ± 61.90
	NGESAT-I	**181.21 ± 24.33**	**190.53 ± 33.59**
	NGESAT-II	**150.28 ± 20.44**	**166.92 ± 23.87**
Kissat-HO	Base	339.54 ± 28.26	447.58 ± 37.46
	Random	378.21 ± 50.68	450.36 ± 57.74
	NGESAT-I	**159.09 ± 19.51**	**177.77 ± 22.14**
	NGESAT-II	**155.53 ± 17.93**	**183.61 ± 23.49**

H Additional Details about training the model in Neural-guided variable selection module under regime (i)

We further clarify the computational cost and generation process for training the model in the neural-guided variable selection module under regime (i). The total enumeration time scales as $n \times 2^k \times T_k$ where T_k is the average solving time of a derived instance given an assignment to k variables.

Candidate k-variable selections are chosen based on the findings of the work[38]: for cryptographic SAT instances, prioritizing the enumeration of variables that reduce second-order terms in the round function to first-order can significantly decrease solving time. For example, in the Simon dataset, prioritizing the reduction of the quadratic term $l_{i+1 \mod n} \cdot l_{i+8 \mod n}$ in the clause: $u_i + l_{i+1 \mod n} \cdot l_{i+8 \mod n} + l_{i+2 \mod n} + r_i + k_i = 0$ to a linear term——by enumerating either $l_{i+1 \mod n}$ or $l_{i+8 \mod n}$ first ——can substantially reduce solving time.

Due to the cyclic symmetry of the round function, the variables form a cyclic sequence of length n, where n denotes the number of key variables. As shown in [38], the effective enumeration strategy is therefore to select a consecutive segment of variables in this cycle, yielding n candidate k-variable selections. Accordingly, our neural-assisted variable selection essentially formulates variable selection as an n-class classification problem. For example, in the Simon-6-16-32 dataset, it is a 16-class classification problem, whereas in the Cipher-12 and Simon-12-32-64 datasets, it is a 32-class classification problem.

In practice, the factor of n can be amortized through parallelization, aligning the data-label generation cost with the testing-time enumeration cost. This makes training the neural-assisted module computationally feasible while preserving the structural effectiveness of candidate variable selections.

Here, we use the Kissat solver to generate labels. Notably, a prediction error made by the model on a single original SAT instance does not affect the correctness of the solution; they only impact the solving time for that instance. As shown in Table 1, our framework adapts across different solvers, consistently achieving substantial performance gains.

I Analysis of the Uncertainty-Driven Strategy in Regime (ii)

Before presenting the formal argument, we first address an empirical observation that motivates the uncertainty-driven design. In our experiments, existing neural approaches—including NeuroSAT [31], CryptoANFNet [46], NeuroBack [40], and related neural assignment predictors—fail to achieve stable test-time prediction accuracy above 50%, even on datasets that are considered favorable to these models. In other words, on a substantial portion of variables, the learned predictors do not perform meaningfully better than random guessing. This phenomenon is not incidental: it reflects the intrinsic difficulty of variables whose assignments cannot be derived through shallow propagation, and it motivates treating near-random predictions as an informative signal rather than a failure mode. The following analysis formalizes this intuition.

(1) Theoretical foundation. The backdoor theory proposed by Williams et al. [42] provides a natural formal framework: there exists a small subset of variables B such that, once assigned, the remaining variables can be solved by a polynomial-time sub-solver, where non-backdoor variables are precisely those that the sub-solver can derive without enumeration-based search. Thus, the relevant partition is between derivable and non-derivable variables, with a subset of the latter constituting the backdoor set B that must be explicitly assigned.

(2) Structural correspondence between the uncertain variable set V_u and backdoor candidates. Under the structural alignment assumption of Regime (ii), p_i remains consistent between training and test instances. Prior studies [37, 31, 40] have demonstrated that GNN-based assignment predictors can effectively learn variables that are determinable via propagation, yet fail on variables that require explicit search. Mojžíšek et al. [25] further proved that GNN-based predictors implicitly perform an optimization analogous to a continuous relaxation, in which variables reachable through short propagation chains receive high-confidence predictions, whereas variables embedded in deep dependency chains degenerate to near-random guessing.

Therefore, under Regime (ii)'s structural alignment assumption where p_i is consistent between training and test instances, the uncertain set $V_u = \{x_i : |p_i - 0.5| < \tau\}$ identifies variables whose propagation depth exceeds the reach—thereby encompassing the backdoor candidates that cannot be derived and must be enumerated.

(3) Variable occurrence frequency (clause participation) quantifies proof complexity reduction. Ben-Sasson et al. [4] proved that resolution proof size is bounded by $\exp(\Omega(w(F) - 1))$, where $w(F)$ denotes the maximum clause width. Atserias et al. [1] and Pipatsrisawat and Darwiche [30]

proved that CDCL polynomially simulates general resolution, thereby linking solving time to proof size. Fixing a variable appearing in f_i clauses reduces the width of all f_i affected clauses, thereby contracting the proof size exponent. Liang et al. [23] empirically confirmed that high-participation variables disproportionately reduce search tree size when prioritized. Thus, ranking V_u by f_i selects variables yielding the greatest proof complexity reduction per enumerated variable.

(4) The top-k variables in V_u ranked by f_i approximate the backdoor set B. The low-accuracy criterion ensures recall: by the arguments in (1)–(2), every backdoor variable $x_i \in B$ is non-derivable, hence $p_i \approx 0.5$ under Regime (ii) assumptions, and consequently $B \subseteq V_u$. The high-participation criterion ensures precision: by the backdoor definition [42], every $x_i \in B$ must trigger sufficient propagation upon assignment, necessitating high f_i. Therefore, the top-k variables in V_u ranked by f_i approximate B.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We summarize the contributions of this paper in both the abstract and the concluding paragraph of the introduction. Each claimed contribution is supported by explicit results presented within the paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the limitations in Section 7 and Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We have provided all the proofs in Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided the dataset settings in Section 5.1 and introduced the setting of each experiment subsection in Section 5.2&5.3 for reproducibility.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We use publicly available datasets, and the related instructions will be released.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We have provided the dataset settings in Section 5.1 and introduced the setting of each experiment subsection in Section 5.2&5.3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We keep the same setting as the previous works.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The detailed information on computation resources is in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conforms, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We have discussed the potential impacts of the paper in Appendix C

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have cited the assets used in the paper.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We have provided the dataset settings in Section 5.1

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.