International Journal of Computer Science and Information Technology
ISSN: 3005-9682 (Print), ISSN: 3005-7140 (Online) | Volume 8, Number 2, Year 2026
DOI: https://doi.org/10.62051/ijcsit.v8n2.08
Journal homepage: https://wepub.org/index.php/IJCSIT/index

GMTSAT: A Neural SAT Solver Based on Multi-Slot Global Updates and Gated Message Passing

Shiyuan Hou *

College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China

*Corresponding Author: houshiyuan@home.hpu.edu.cn

ABSTRACT

The Boolean Satisfiability Problem (SAT) is a fundamental NP-complete problem in computer science, with broad applications in formal verification, electronic design automation, and combinatorial optimization. In recent years, neural SAT solvers based on message-passing neural networks (MPNNs) have shown promise in variable assignment prediction and heuristic guidance. However, their reliance on local graph topology severely limits their ability to capture long-range logical dependencies. To address this limitation, this paper proposes GMTSAT, a novel neural SAT solving architecture that integrates multi-slot global attention with gated message passing to enhance global reasoning capability and representation robustness. Specifically, we introduce a set of learnable global latent slots that are independent of the underlying graph topology. Through Transformer-based cross-attention and self-attention mechanisms, these slots enable single-step global information aggregation and broadcast across the entire graph while preserving permutation invariance. In addition, a gated residual perceptron is designed to replace conventional MLPs in the message-passing process, effectively suppressing noisy signals and improving training stability. Extensive experiments on large-scale random 3-SAT benchmarks demonstrate that GMTSAT significantly outperforms strong baselines such as NeuroSAT and TG-SAT in variable assignment prediction accuracy, while exhibiting superior scalability to larger problem instances. Ablation studies confirm the independent effectiveness and complementary benefits of the proposed global slot mechanism and gating modules. Furthermore, joint experiments with the state-of-the-art CDCL solver Kissat systematically evaluate two application paradigms: a hard-constraint strategy that aggressively fixes high-confidence variables to reduce the search space, and a soft-constraint guidance strategy that preserves solver completeness while achieving consistent speedups. The results indicate that GMTSAT provides an effective and scalable paradigm for deep integration between neural reasoning models and traditional SAT solvers.

KEYWORDS

Boolean satisfiability; Neural SAT solver; Graph neural networks; Global attention; Heuristic guidance; Kissat

1. INTRODUCTION

The Boolean Satisfiability Problem (SAT) [1] is a fundamental NP-complete problem in computer science [2] and has been widely applied in formal verification [3], electronic design automation (EDA) [4], and combinatorial optimization [5]. Modern SAT solvers, most notably those based on Conflict-Driven Clause Learning (CDCL) [6], have achieved remarkable success on large-scale industrial instances. However, when confronted with instances of extremely high complexity, their heuristic-driven search procedures still exhibit exponential runtime growth.

In recent years, data-driven neural SAT solvers have emerged as an active research direction. These approaches aim to leverage deep learning models to directly predict variable assignments or to guide branching heuristics in traditional solvers. Among them, graph neural network (GNN)-based methods [7] have become particularly prominent. Representative neural solvers such as NeuroSAT [8] model formulas in conjunctive normal form (CNF) as literal–clause bipartite graphs and iteratively update node embeddings using message passing neural networks (MPNNs).

Despite their effectiveness, such locally driven update mechanisms suffer from inherent limitations in capturing long-range dependencies. In SAT instances with deep logical chains, two variables that are strongly correlated at the logical level may be separated by a large distance in the graph topology. Relying solely on local message passing requires information to traverse many iterations before reaching distant nodes, a process that is often accompanied by information attenuation and over-smoothing. As a result, the model struggles to establish globally consistent assignment strategies. Furthermore, conventional multilayer perceptrons (MLPs) lack effective mechanisms for filtering noisy messages when processing complex graph interactions, which further constrains solver performance.

To address these limitations, this paper proposes an enhanced neural SAT solving architecture. Building upon the strengths of long short-term memory (LSTM) networks in modeling local topological interactions, we introduce a Multi-Slot Global Update mechanism. Specifically, the model initializes a set of learnable global latent vectors (referred to as slots) that are independent of the underlying graph structure. Through cross-attention mechanisms in a Transformer architecture [9], these slots aggregate features from all graph nodes and act as hubs for long-range information exchange, enabling single-step global communication between arbitrary nodes. In addition, we design a Gated Residual MLP to replace conventional feed-forward networks, where gating mechanisms dynamically regulate information flow, thereby enhancing robustness in nonlinear feature extraction.

The main contributions of this work are summarized as follows:

(1) We propose a hybrid graph neural network architecture that introduces multi-slot global latent vectors, effectively addressing the long-range dependency modeling challenges faced by traditional MPNNs in SAT solving.

(2) We design a gated and residual feature extraction module that improves representational capacity and training stability during message passing.

2. RELATED WORK

2.1. Traditional Solvers and Early Machine Learning Approaches

SAT solving techniques have been developed over several decades. Most state-of-the-art complete solvers in industrial applications are built upon the CDCL conflict analysis framework. By integrating non-chronological backtracking [10], Boolean constraint propagation (BCP), and heuristic branching strategies such as VSIDS [11], CDCL solvers can efficiently handle structured instances containing millions of variables. Modern solvers such as Kissat [12] and MiniSat [13] fall into this category. Nevertheless, despite their strong performance, CDCL solvers rely heavily on manually designed heuristics and are difficult to adaptively optimize for specific problem distributions.

Prior to the widespread adoption of deep learning, researchers explored classical machine learning methods to predict SAT satisfiability or solver runtime [14]. For instance, SATzilla [15] selects the most suitable solver for a given instance based on handcrafted feature vectors, while Devlin et al. transformed SAT solving into a binary classification task using manually engineered features [16]. Although these approaches achieved certain improvements, their performance strongly depends on the quality of hand-designed features, which are limited in capturing deep structural information and often exhibit poor generalization.

2.2. Graph Neural Network-Based Neural Solvers

With advances in deep learning, end-to-end neural SAT solvers have become a major research focus. Early neural approaches attempted to process CNF formulas as text using RNNs or CNNs, but failed to respect the permutation invariance inherent in Boolean formulas. The introduction of NeuroSAT marked a pivotal milestone in this field. By modeling CNF formulas as permutation-invariant literal–clause bipartite graphs and applying MPNNs to iteratively update node embeddings, NeuroSAT demonstrated that neural networks could learn solution-relevant structural properties using only a single-bit supervision signal (satisfiable/unsatisfiable), and could even decode concrete variable assignments.

Subsequent work largely followed this paradigm. For example, Selsam and Bjørner employed GNNs to predict unsatisfied cores to guide CDCL solvers [17] while NLocalSAT used neural networks to optimize the initialization of local search procedures [18]. However, most of these methods retain the original local message passing mechanism, where nodes interact only with their immediate neighbors. When applied to instances with high logical depth, such localized updates limit the efficiency of constraint propagation and often lead to information dissipation in deep networks.

2.3. Hybrid Architectures and Attention-Based Models

To overcome the inherent limitations of MPNNs in capturing long-range dependencies, recent studies have explored incorporating Transformer architectures or global information aggregation mechanisms into SAT solving. One line of work applies Transformers directly to graph-structured data. Shi et al. proposed SATformer [19], which employs a hierarchical Transformer to capture correlations among clauses, particularly for learning minimal unsatisfiable cores. More recently, Chang et al. introduced the TG-SAT framework [20], which integrates gated recurrent units (GRUs) into a Transformer architecture to aggregate multi-hop neighborhood information and leverages cross-attention to optimize interactions between literals and clauses.

These results indicate that hybrid architectures combining sequential models (e.g., GRUs or LSTMs) with attention mechanisms are well suited for complex logical reasoning tasks. However, existing hybrid approaches such as TG-SAT primarily focus on enhancing direct interactions among nodes and still lack a global working memory that is independent of the graph topology to explicitly store and broadcast global conflict states. Motivated by this observation, we propose a novel architecture based on global latent slots, which establishes a set of learnable, topology-independent hubs that fundamentally break graph-distance constraints and enable single-step global aggregation and dissemination of information across the entire graph.

3. METHODOLOGY

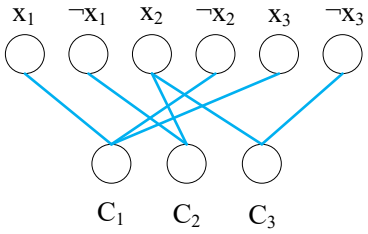
Our model is built upon the message passing neural network (MPNN) framework. By incorporating multi-slot global latent vectors and gating mechanisms, we construct a hybrid update system capable of effectively modeling long-range dependencies in SAT instances.

3.1. Problem Definition and Graph Representation

In the SAT problem, Boolean formulas are typically expressed in conjunctive normal form (CNF). A CNF formula is a conjunction of multiple clauses, where each clause is a disjunction of literals. A literal represents either a Boolean variable or its negation. For example, $\phi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (x_2 \vee \neg x_3)$ where ϕ is a CNF formula consisting of three clauses, x_i denotes a Boolean variable, and $\neg x_i$ denotes its negated form.

Let the variable set be $V = \{x_1, x_2, \ldots, x_n\}$ and the clause set be $\phi = \{C_1, C_2, \ldots, C_m\}$, where each clause C_i is a disjunction of literals. The SAT problem can then be formally defined as determining whether there exists a Boolean assignment function $f: V \to \{0,1\}$ such that $C_i(f) = 1$ for all clauses $C_i \in \phi$ If such an assignment exists, the formula is satisfiable (SAT); otherwise, it is unsatisfiable (UNSAT). In practical implementations, SAT instances are typically stored in DIMACS format. For neural model training, the DIMACS representation is further transformed into a clause–literal incidence matrix M.

Specifically, a CNF formula ϕ is modeled as a bipartite graph $G = (\mathcal{V}_L \cup \mathcal{V}_C, \mathcal{E})$ The modeling strategy and its corresponding mathematical formulation are illustrated in Fig. 1. Here, $\mathcal{V}_L$ denotes the set of literal nodes (including both positive and negative literals), and $\mathcal{V}_C$ denotes the set of clause nodes. An edge $(l, c) \in \mathcal{E}$ exists if and only if literal l appears in clause c . The input to the model consists of the adjacency matrix of this bipartite graph together with initialized node features. In the clause–literal incidence matrix, each row corresponds to a clause, and each column corresponds to a literal in the set $\{x_1, \ldots, x_n, \neg x_1 \ldots, \neg x_n\}$ the learning objective is to predict a Boolean assignment $y_i \in \{0,1\}$ or each variable x_i such that the formula ϕ is satisfied. The bipartite graph representation and the corresponding incidence matrix of the example formula ϕ are shown in Fig. 1.



$$M = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figure 1. Bipartite graph representation and incidence matrix M corresponding to the CNF formula ϕ

3.2. Overall Architecture

The proposed model adopts an iterative hybrid update architecture consisting of three main stages: initialization, T rounds of core iterative updates, and final readout for prediction. Let $\mathrm{L}^{(t)} \in \mathbb{R}^{2n \times d}$ and $\mathrm{C}^{(t)} \in \mathbb{R}^{m \times d}$ denote the literal state matrix and clause state matrix at iteration t, respectively, where d is the hidden dimension. To enable global information exchange beyond the graph topology, we introduce a set of learnable global latent slots $\mathrm{S}^{(t)} \in \mathbb{R}^{k \times d}$ which are independent of the underlying graph structure. Here, k denotes the number of slots and is set to 8 in this work. The detailed data flow and module composition of the proposed architecture are illustrated in Fig. 2.

At each iteration t, the model performs the following operations sequentially:

Global Information Broadcasting: The global latent slots $\mathrm{S}^{(t-1)}$ are injected into local node representations to disseminate global contextual information.

Local Topological Update: Literal and clause states $\mathrm{C}^{(t)}$ and $\mathrm{L}^{(t)}$ are updated via gated message passing combined with LSTM-based state transitions, capturing local structural dependencies.

Global State Evolution: The global latent slots $\mathrm{S}^{(t)}$ are updated using a Transformer module to aggregate and evolve global information across iterations.

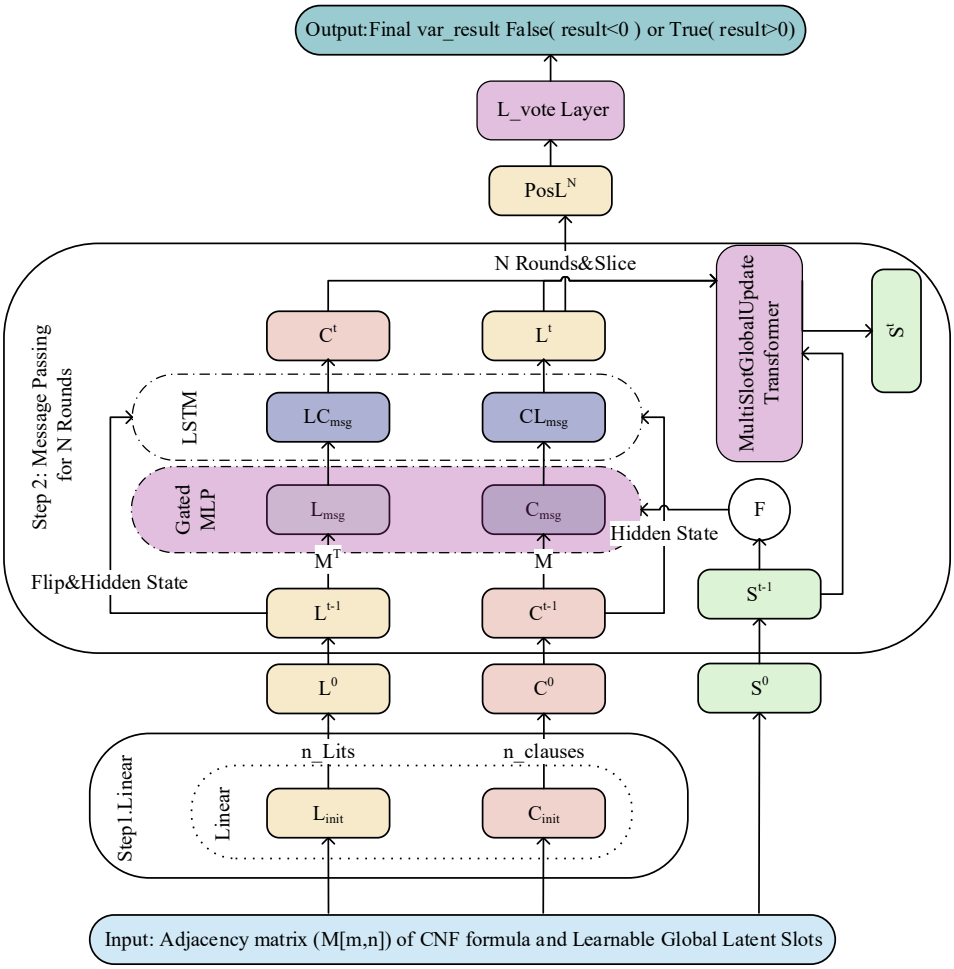


Figure 2. Overall architecture of the proposed GMTSAT model

3.3. Enhanced Gated Message Passing

To mitigate the impact of noisy features commonly encountered in conventional MLP-based transformations, we design a Gated Residual Perceptron (GRP) as the message generation function. Given an input feature vector x, the output of the gated transformation $\mathcal{F}_{gate}(x)$ is defined as follows:

$$h = \mathrm{GELU}(\mathrm{LN}(W_1 x)) \tag{1}$$

$$g = \sigma(W_g x) \tag{2}$$

$$F_{gate}(x) = (W_2 h) \odot g + x_{res} \tag{3}$$

Where σ denotes the sigmoid activation function, ⊙ represents element-wise multiplication, and LN denotes layer normalization. The gating coefficient g dynamically regulates the information flow, allowing the network to selectively suppress noisy or less informative features. The residual term x_{res} preserves the original input and is set to zero when the input and output dimensions do not match. The detailed gating and residual connection mechanism is illustrated in Fig. 3.

The message passing updates from literals to clauses (L→C) and from clauses to literals (C→L) are performed as follows:

First, the global latent slots from the previous iteration, $S^{(t-1)}$, are linearly projected and broadcast to all nodes, producing a global context vector denoted as s_{ctx}. For clause updates, the local clause states are concatenated with the global context to generate and aggregate incoming messages:

$$\mathbf{M}_{L\to C} = \mathbf{A}^T \cdot \mathrm{F}_{gate}([\mathbf{L}^{(t-1)} \,||\, \mathbf{s}_{ctx}]) \tag{4}$$

$$(\mathbf{c}_{state}^{(t)}, \mathbf{C}^{(t)}) = \mathrm{LSTM}_C(\mathbf{M}_{L\to C}, \mathbf{c}_{state}^{(t-1)}) \tag{5}$$

where A denotes the sparse clause–literal adjacency matrix, and || represents feature concatenation.

Subsequently, the updated clause states are propagated back to literal nodes. Notably, literal updates explicitly incorporate variable negation (flip) information, ensuring that messages from a negated literal ¬x are correctly transmitted to its corresponding positive literal x. Let L_{flip} denote the flipped literal representation; the literal update is given by:

$$\mathbf{M}_{C\to L} = \mathbf{A} \cdot \mathrm{F}_{gate}([\mathbf{C}^{(t)} \,||\, \mathbf{s}_{ctx}]) \tag{6}$$

$$(\mathbf{l}_{state}^{(t)}, \mathbf{L}^{(t)}) = \mathrm{LSTM}_L([\mathbf{M}_{C\to L} \,||\, \mathrm{flip}(\mathbf{L}^{(t-1)})], \mathbf{l}_{state}^{(t-1)}) \tag{7}$$

This update scheme ensures effective constraint propagation within the local graph topology while simultaneously incorporating global guidance from the latent slots, thereby enabling more coherent and globally consistent message passing dynamics.

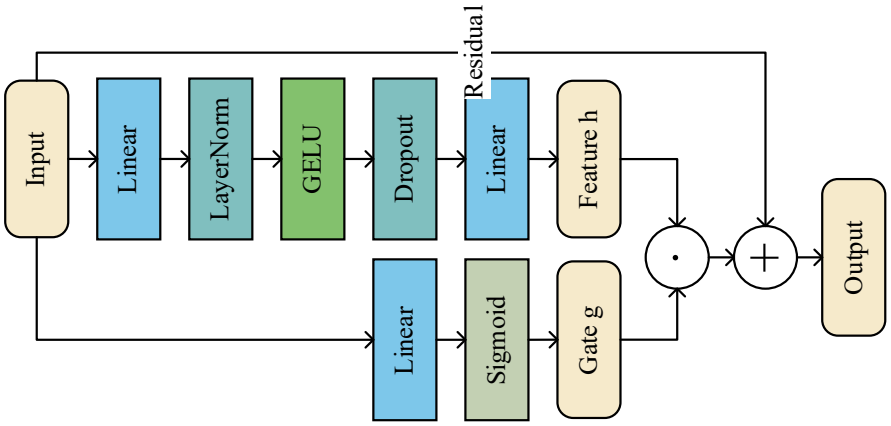


Figure 3. Internal structure of the gated residual perceptron

3.4. Multi-Slot Global Update Mechanism

To effectively capture long-range dependencies, we propose a multi-slot global Transformer module, which is responsible for updating the global state at each iteration, as illustrated in Fig. 4.

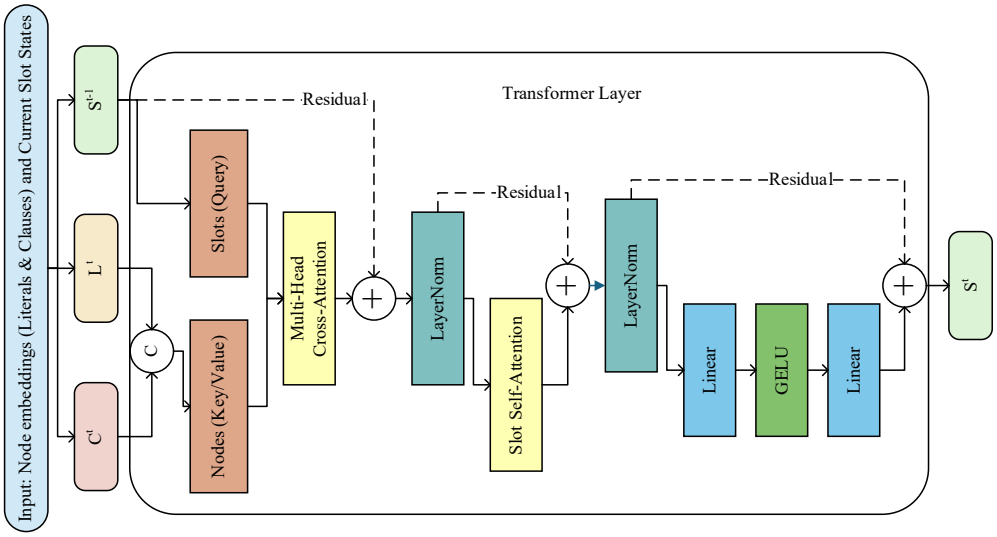


Figure 4. Detailed update mechanism of the multi-slot global Transformer layer

In this module, the k global latent slots are treated as query vectors, while all nodes in the graph serve as keys and values. Let $X_{nodes} = [L^{(t)}; C^{(t)}] \in \mathbb{R}^{(2n+m)\times d}$ denote the collection of all node representations at iteration t, The update of the global state $S^{(t)}$ consists of two successive attention stages.

1 Cross-Attention: Information Aggregation from Graph Nodes

$$\mathbf{Q} = \mathbf{S}^{(t-1)}, \quad \mathbf{K} = \mathbf{V} = \mathbf{X}_{nodes} \tag{8}$$

$$\mathbf{S}_{mid} = \text{LayerNorm}(\mathbf{S}^{(t-1)} + \text{MultiHeadAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V})) \tag{9}$$

This operation enables each slot to attend to structurally or semantically relevant regions of the graph regardless of their topological distance, thereby facilitating single-step global information aggregation.

2 Self-Attention: Slot-to-Slot Coordination

In the second stage, self-attention is applied among the slots to allow mutual interaction and coordination:

$$\mathbf{S}^{(t)} = \text{LayerNorm}(\mathbf{S}_{mid} + \text{MultiHeadAttn}(\mathbf{S}_{mid}, \mathbf{S}_{mid}, \mathbf{S}_{mid})) \tag{10}$$

The self-attention mechanism enables information exchange and coordination among different slots, encouraging a globally consistent solving strategy. The updated slot representations are subsequently processed by a feed-forward network (FFN) to obtain $S^{(t)}$

3.5. Prediction and Training Objective

After N iterations, we extract the representations corresponding to positive literals from the final literal state matrix $L^{(N)}$ denoted as $L_{pos} \in \mathbb{R}^{n\times d}$. The prediction head consists of a multilayer perceptron (MLP) equipped with layer normalization and GELU activation, which maps the high-dimensional features to scalar logits:

$$\hat{\mathbf{y}} = \mathbf{W}_{out}(\text{GELU}(\text{LN}(\mathbf{W}_{in}\mathbf{L}_{pos}))) \tag{11}$$

A positive logit $\hat{y}_i > 0$ indicates that the variable x_i is predicted to be true, while a negative value indicates false. The model is trained in a supervised learning setting, where ground-truth assignments$\mathbf{y}^{gt} \in \{0,1\}^n$ are obtained from a conventional SAT solver. The training objective is the binary cross-entropy (BCE) loss. To ensure effective learning, we adopt a SAT-only supervision strategy, in which the loss is computed exclusively on satisfiable (SAT) instances, while unsatisfiable (UNSAT) instances are filtered out via a masking mechanism:

$$\mathrm{L} = \frac{1}{\sum \mathrm{I}(F_j \in \mathrm{SAT})} \sum_j \mathrm{I}(F_j \in \mathrm{SAT}) \cdot \mathrm{BCE}(\hat{\mathbf{y}}_j, \mathbf{y}_j^{gt}) \tag{12}$$

This strategy forces the model to focus on learning meaningful variable assignment distributions rather than merely performing SAT/UNSAT classification.

4. EXPERIMENTS AND ANALYSIS

In this section, we conduct large-scale benchmark evaluations to assess the effectiveness of the proposed neural SAT solver in terms of variable prediction accuracy and its ability to accelerate a conventional SAT solver (Kissat). The experiments focus on comparing two model application strategies: direct variable fixing (addVar) and heuristic branching guidance (Guidance).

4.1. Training Dataset

Training deep learning models requires a sufficiently large amount of data. Considering the characteristics of the Boolean satisfiability problem, all datasets used in our experiments consist of randomly generated 3-SAT instances in CNF format. Random 3-SAT problems are selected as the core experimental dataset for the following reasons.

First, the difficulty of SAT solving increases exponentially with the number of variables. Random 3-SAT instances not only allow for efficient generation of samples at different scales, but also provide a convenient mechanism to control problem difficulty through the ratio between the number of clauses and the number of variables. In this work, we fix this ratio to 4.26, motivated by the well-known phase transition phenomenon in random 3-SAT problems.

Specifically, when the clause-to-variable ratio deviates from 4.26, instances tend to become predominantly satisfiable or unsatisfiable, resulting in relatively easier solving scenarios. At the critical ratio of 4.26, however, the satisfiability probability drops sharply to approximately 50%, producing instances that are empirically the hardest to solve. During dataset generation, the relationship between the number of clauses m and the number of variables n follows the formulation given in Eq 13:

$$m = 4.26n + 58.26n^{-\frac{2}{3}} \tag{13}$$

The dataset used in this work is a publicly available benchmark released in [21]. This dataset is selected primarily because of its public accessibility, which ensures a unified standard for reproducibility and fair comparison in training and evaluation. In addition, generating large-scale SAT instances and converting them into the model-specific input format is computationally expensive. On our training platform, generating instances with 300 variables and preprocessing them for model training requires approximately one week. Since the required time increases multiplicatively with the number of variables, we adopt this existing dataset to ensure feasibility and consistency. The dataset contains instances with the number of variables ranging from 100 to 450, increasing in steps of 50. The proportion of satisfiable and unsatisfiable instances is controlled to be 1:1. For each problem

scale, the dataset consists of 10,000 CNF formulas, which are split into training, validation, and test sets with a ratio of 8:1:1.

4.2. Training Setup

We compare the proposed GMTSAT model with the standard baseline NeuroSAT in terms of training accuracy. It is worth noting that NeuroSAT is the seminal deep learning approach for SAT solving and serves as a widely recognized benchmark. Since the original NeuroSAT formulation produces a binary satisfiable/unsatisfiable prediction, we adapt its output layer for a fair comparison. Specifically, without modifying the original training logic of NeuroSAT, we adjust only the final prediction stage by removing the aggregation and averaging over positive and negative literals, thereby enabling per-variable assignment prediction. The same adaptation strategy is applied to TG-SAT.

All models are trained and evaluated under a unified parameter configuration. The embedding dimension is set to d=128, the number of attention heads in the Transformer module is 4, and the number of message passing iterations is fixed to N=26. We use a learning rate of 1e-4 and train all models for 50 epochs. All experiments are conducted on a single NVIDIA RTX 4090 GPU using PyTorch version 2.4.1 with CUDA 11.8. The implementations of NeuroSAT, TG-SAT, and the Kissat solver, along with the experimental configurations, are made publicly available in the GitHub repository associated with this paper.

4.3. Model Training and Evaluation

To evaluate the generalization ability and solving performance of the models across different problem scales, we conduct comparative experiments on GMTSAT and the baseline models NeuroSAT and TG-SAT over eight datasets with the number of variables $N \in \{100,150,\ldots,450\}$ Fig. 5 illustrates the prediction accuracy of variable assignments achieved by the three models on the test sets across different problem sizes.

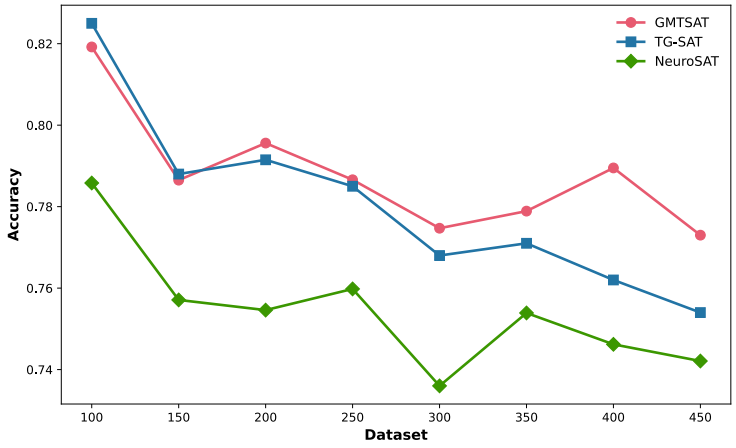


Figure 5. Comparison of variable assignment prediction accuracy between GMTSAT and NeuroSAT on datasets with 100–450 variables

The experimental results show that NeuroSAT achieves the lowest overall performance among the evaluated models, highlighting the inherent limitations of pure message passing mechanisms in modeling long-range dependencies within deep graph structures. Across all problem scales, GMTSAT consistently outperforms the baseline NeuroSAT, demonstrating the effectiveness of the proposed global update mechanism. Moreover, when evaluated on larger and more complex instances, GMTSAT also exhibits superior scalability and robustness compared to the hybrid architecture TG-SAT. On smaller-scale instances, TG-SAT achieves a slight advantage, which can be attributed to its fine-grained global attention mechanism. When the number of nodes is limited, full-graph attention can effectively capture relevant features while introducing relatively little noise. However, as the

problem scale increases, the advantages of GMTSAT's multi-slot global update mechanism become increasingly evident. In the range of N=200 to N=300, GMTSAT consistently surpasses TG-SAT in prediction accuracy. For larger instances, TG-SAT exhibits a noticeable performance degradation, suggesting that standard Transformer-based architectures are more susceptible to interference from irrelevant nodes in large graph structures. In addition, TG-SAT incurs substantially higher training costs due to its O(NlogN) time complexity. In practice, when the number of variables reaches 400, the training time of TG-SAT becomes approximately six times longer than that of GMTSAT. By introducing the multi-slot global update mechanism, GMTSAT effectively alleviates information over-smoothing and enhances the modeling of long-range dependencies. As a result, it maintains a clear performance advantage even on the most challenging instances with N=450. Furthermore, GMTSAT consistently demonstrates stable superiority across different data distributions, validating its stronger feature representation capability and generalization performance. These properties make GMTSAT particularly well suited for providing reliable initial variable assignments to downstream SAT solvers.

4.4. Experimental Analysis

Based on the trained models, we first evaluate the hard-constraint variable fixing strategy. Under this strategy, variables whose predicted confidence pi exceeds a predefined threshold τ are directly fixed by adding corresponding unit clauses to the original CNF formula. When injecting unit clauses, it is necessary to update the total variable and clause counts in the preamble of the DIMACS file accordingly; otherwise, the Kissat solver will report an inconsistency error during parsing.

We conduct experiments across multiple datasets using different confidence thresholds. Since the effectiveness of variable fixing depends on both the quality of the trained model and the computational environment, and because NeuroSAT achieves substantially lower prediction accuracy than GMTSAT under identical settings, we report results obtained using GMTSAT only, evaluated under a unified configuration across different thresholds and problem scales. The reported results are selected from multiple experimental runs and are representative of the overall trends observed.

The experimental results reveal a clear and consistent pattern across all datasets. As the total number of variables increases, the number of variables fixed by the model decreases approximately linearly with increasing confidence threshold τ. In particular, for the datasets with 100 and 150 variables, when the threshold is set to τ=0.99, no variables satisfy the confidence requirement, and consequently, no unit clauses are added. In this case, the modified solver exhibits nearly identical runtime behavior to the original Kissat solver, indicating that the model introduces no adverse effects when insufficiently confident. The distribution of the proportion of fixed variables across different datasets and confidence thresholds is summarized in Fig. 6 and Table 1.

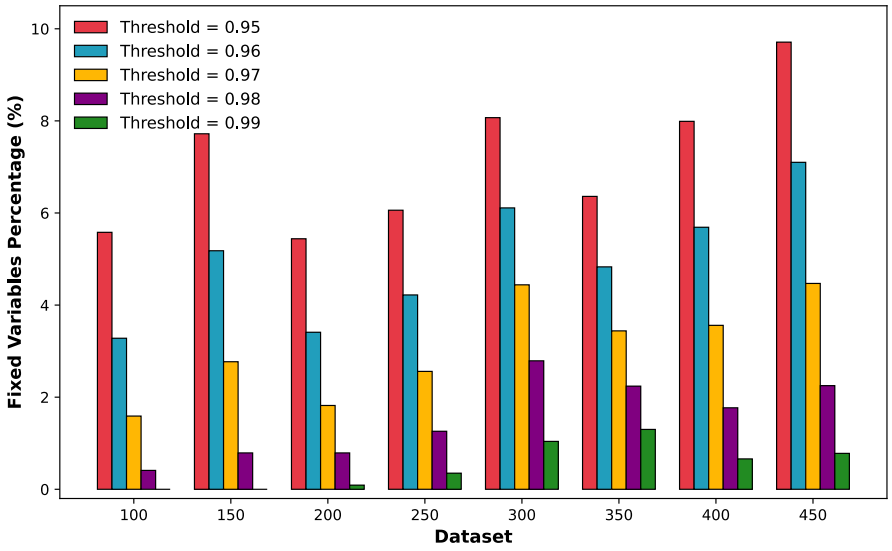


Figure 6. Percentage of variables fixed under different confidence thresholds across datasets

Table 1. Percentage of variables fixed under different confidence thresholds across datasets

Dataset	Number of Variables	Prediction Accuracy				
		>0.95(%)	>0.96(%)	>0.97(%)	>0.98(%)	>0.99(%)
100	100000	5.58	3.28	1.59	0.41	0.00
150	150000	7.72	5.18	2.77	0.79	0.00
200	200000	5.44	3.41	1.82	0.79	0.09
250	250000	6.06	4.22	2.56	1.26	0.35
300	300000	8.07	6.11	4.44	2.79	1.04
350	350000	6.36	4.83	3.44	2.24	1.30
400	400000	7.99	5.69	3.56	1.77	0.66
450	450000	9.71	7.10	4.47	2.25	0.78

4.4.1. Performance Evaluation of Direct Variable Fixing

Under lower confidence thresholds, the model fixes a larger number of variables, resulting in more pronounced solver acceleration. The trade-off between speedup and prediction error under different thresholds is illustrated in Fig. 7.

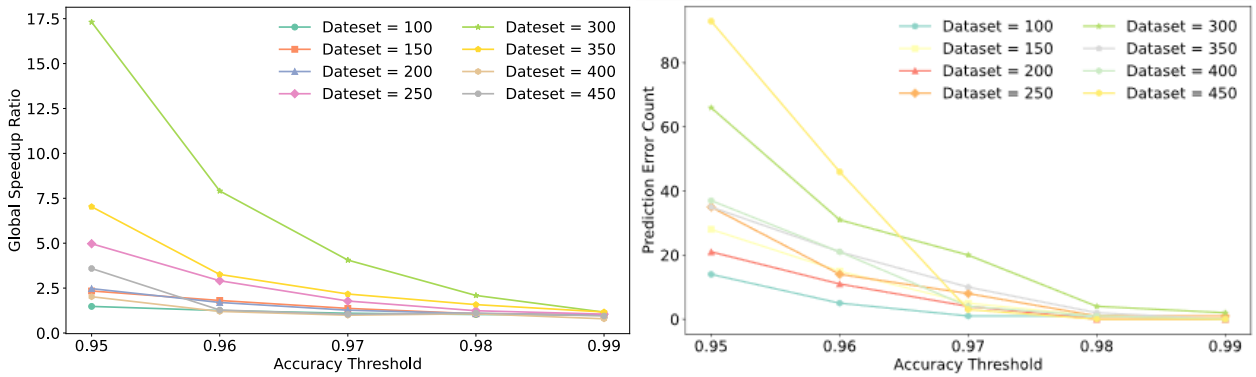


Figure 7. Analysis of global speedup (left) and prediction error count (right) under different confidence thresholds using the addVar strategy

On the dataset with 300 variables, when the confidence threshold is set to τ=0.95, the global speedup reaches 17.31×, while the number of conflicts generated by the solver is reduced by 91.05%. This

observation indicates that the model successfully identifies and fixes a subset of backbone variables, effectively transforming the original combinatorial search process into a much simpler Boolean constraint propagation (BCP) procedure. However, due to the tightly coupled structure of SAT formulas, the addition of incorrect unit clauses can significantly alter the original solution space and potentially render the instance unsatisfiable. Consequently, prediction errors introduced by aggressive variable fixing can be detrimental to solver performance. As expected, the error rate decreases as the confidence threshold increases. In our experiments, when the threshold is set to $\tau = 0.98$, the probability of incorrect predictions drops below 0.1% in most cases, while the solver still benefits from a substantial improvement in efficiency.

Detailed comparisons of the variable fixing strategy across different datasets are reported in Tables 2–9. Notably, although both satisfiable (SAT) and unsatisfiable (UNSAT) instances are included in the overall dataset, only SAT instances are used during model training. Specifically, during preprocessing, CNF formulas labeled as SAT are transformed according to their satisfiability annotations and used for supervised training of variable polarity prediction. Since UNSAT instances do not admit any valid variable assignment, they cannot be directly utilized for supervised assignment prediction and are therefore excluded from the training process. During testing, however, the satisfiability status of the CNF formulas is unknown. Consequently, the trained model produces variable polarity predictions for all instances, including UNSAT ones. Empirically, we observe that for these UNSAT instances, the Kissat solver may fail to terminate within the official SAT Competition time limit of 5000 seconds when run alone. In contrast, after applying model-guided variable fixing, Kissat is often able to quickly determine unsatisfiability, achieving a 100% correctness rate on these instances. For example, on the dataset with 350 variables, the proposed strategy successfully solved 129 UNSAT instances that originally timed out when using the vanilla Kissat solver. This result demonstrates that the model is capable of breaking search stagnation and significantly improving solver robustness on hard UNSAT cases. We hypothesize that the trained model implicitly captures structural signals related to the satisfiability core of CNF formulas. Similar observations have been reported in NeuroCore, where learned representations encode information indicative of satisfiability and conflict structure. Our experimental findings are consistent with this perspective, further supporting the effectiveness of learning-based guidance in SAT solving.

Table 2. Kissat solver comparison on 100-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	14	5	1	1	0
Double Timeouts	0	0	0	0	0
Additional Solved Instances	0	0	0	0	0
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	8	24	47	88	132
Total Slowdown Time	0.08	0.24	0.47	0.88	1.32
Average Slowdown Time	0.01	0.01	0.01	0.01	0.01
Maximum Slowdown Time	0.01	0.01	0.01	0.01	0.01
Effective Instances	986	995	999	999	1000
Original Total Runtime	37.4900	36.4500	37.6900	37.7300	36.6300
Simplified Total Runtime	25.3600	29.1000	34.1100	36.7500	36.4300
Overall Speedup	1.48x	1.25x	1.10x	1.03x	1.01x
Original Conflicts	608590	612697	614499	614499	614580
Simplified Conflicts	298337	415599	522900	596253	614580
Conflict Reduction Rate	-50.98%	-32.17%	-14.91%	-2.97%	-0.00%

Table 3. Kissat solver comparison on 150-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	28	15	5	1	0
Double Timeouts	0	0	0	0	0
Additional Solved Instances	0	0	0	0	0
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	6	37	100	277	296
Total Slowdown Time	0.06	0.57	1.96	5.82	4.31
Average Slowdown Time	0.01	0.02	0.02	0.02	0.01
Maximum Slowdown Time	0.01	0.05	0.12	0.14	0.05
Effective Instances	972	985	995	999	1000
Original Total Runtime	90.1500	91.5300	94.4600	93.1900	93.3600
Simplified Total Runtime	38.5300	50.6200	68.8100	86.8400	92.8400
Overall Speedup	2.34x	1.81x	1.37x	1.07x	1.01x
Original Conflicts	3404102	3450241	3470766	3493169	3494368
Simplified Conflicts	970730	1574746	2403227	3255274	3494368
Conflict Reduction Rate	-71.48%	-54.36%	-30.76%	-6.81%	-0.00%

Table 4. Kissat solver comparison on 200-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	21	11	4	0	0
Double Timeouts	0	0	0	0	0
Additional Solved Instances	0	0	0	0	0
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	64	158	297	364	422
Total Slowdown Time	3.13	12.57	28.48	37.55	26.72
Average Slowdown Time	0.05	0.08	0.10	0.10	0.06
Maximum Slowdown Time	0.33	0.92	0.75	0.98	0.77
Effective Instances	979	989	996	1000	1000
Original Total Runtime	464.0897	467.2407	470.7648	475.5169	472.6744
Simplified Total Runtime	187.7782	274.4117	369.2599	439.3741	470.3787
Overall Speedup	2.47x	1.70x	1.27x	1.08x	1.00x
Original Conflicts	15452368	15576268	15696871	15760630	15760630
Simplified Conflicts	6579923	9537469	12701974	14871118	15738199
Conflict Reduction Rate	-57.42%	-38.77%	-19.08%	-5.64%	-0.14%

Table 5. Kissat solver comparison on 250-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	35	14	8	1	1
Double Timeouts	0	0	0	0	0
Additional Solved Instances	0	0	0	0	0
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	87	132	198	298	399
Total Slowdown Time	15.44	43.11	94.06	192.54	213.92
Average Slowdown Time	0.18	0.33	0.48	0.65	0.54
Maximum Slowdown Time	1.43	3.12	4.60	6.63	6.97
Effective Instances	965	986	992	999	999
Original Total Runtime	3450.5973	3505.0799	3534.2524	3561.2039	3575.6518
Simplified Total Runtime	694.2526	1203.3486	1987.0878	2869.1277	3420.0408
Overall Speedup	4.97x	2.91x	1.78x	1.24x	1.05x
Original Conflicts	81911301	83218563	83648801	84027178	84027178
Simplified Conflicts	20490485	33332483	51463276	70699816	82338110
Conflict Reduction Rate	-74.98%	-59.95%	-38.48%	-15.86%	-2.01%

Table 6. Kissat solver comparison on 300-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	66	31	20	4	2
Double Timeouts	0	0	0	0	0
Additional Solved Instances	1	2	1	1	1
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	88	131	168	205	386
Total Slowdown Time	51.60	156.86	282.48	490.01	1721.30
Average Slowdown Time	0.59	1.20	1.68	2.39	4.46
Maximum Slowdown Time	6.05	25.46	19.40	35.98	67.54
Effective Instances	933	967	979	995	997
Original Total Runtime	36032.0641	35858.8210	33817.6399	29834.9457	30219.3584
Simplified Total Runtime	2081.5485	4535.0961	8327.2098	14275.5713	25997.2569
Overall Speedup	17.31x	7.91x	4.06x	2.09x	1.16x
Original Conflicts	516055054	521778566	527054541	529269437	530224512
Simplified Conflicts	46174674	91918130	166271972	296467680	477502895
Conflict Reduction Rate	-91.05%	-82.38%	-68.45%	-43.99%	-9.94%

Table 7. Kissat solver comparison on 350-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	35	21	10	2	0
Double Timeouts	7	51	126	247	347
Additional Solved Instances	388	387	297	159	61
Simplified Timeouts	0	2	3	2	6
Number of Slowdown Cases	129	158	188	217	253
Total Slowdown Time	478.78	921.94	1149.27	2191.73	2590.10
Average Slowdown Time	3.71	5.84	6.11	10.10	10.24
Maximum Slowdown Time	64.00	97.90	75.85	236.97	122.06
Effective Instances	570	539	564	590	586
Original Total Runtime	30123.6430	20178.0010	23560.6928	28387.6456	26566.4214
Simplified Total Runtime	4284.8447	6184.4075	10878.6156	17936.7542	22737.3365
Overall Speedup	7.03x	3.26x	2.17x	1.58x	1.17x
Original Conflicts	397015287	238144127	294660946	362659988	339003372
Simplified Conflicts	85008234	96347414	163531346	258085412	301746866
Conflict Reduction Rate	-78.59%	-59.54%	-44.50%	-28.84%	-10.99%

Table 8. Kissat solver comparison on 400-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	37	21	4	1	0
Double Timeouts	27	124	229	281	284
Additional Solved Instances	257	161	55	3	0
Simplified Timeouts	3	4	6	2	0
Number of Slowdown Cases	203	268	291	334	367
Total Slowdown Time	1425.96	4739.25	5927.80	6396.57	7304.07
Average Slowdown Time	7.02	17.68	20.37	19.15	19.90
Maximum Slowdown Time	279.41	510.44	479.00	1119.23	1067.20
Effective Instances	676	690	706	713	716
Original Total Runtime	6478.3611	10840.8556	11246.1336	14942.3708	10511.0345
Simplified Total Runtime	3214.4750	9053.8170	11366.3771	14007.1072	13367.9611
Overall Speedup	2.02x	1.20x	0.99x	1.07x	0.79x
Original Conflicts	92022146	103066899	126222707	130627353	139327858
Simplified Conflicts	50890435	86181059	124166735	122927296	157704233
Conflict Reduction Rate	-44.70%	-16.38%	-1.63%	-5.89%	+13.19%

Table 9. Kissat solver comparison on 450-variable dataset with varying confidence thresholds

Performance Metrics	Prediction Accuracy				
	>0.95	>0.96	>0.97	>0.98	>0.99
Total Instances	1000	1000	1000	1000	1000
Prediction Errors	93	46	3	0	0
Double Timeouts	1	145	447	500	500
Additional Solved Instances	499	355	53	0	0
Simplified Timeouts	0	18	19	7	1
Number of Slowdown Cases	88	143	187	220	236
Total Slowdown Time	627.91	3687.16	5792.05	4747.69	5694.39
Average Slowdown Time	7.14	25.78	30.97	21.58	24.13
Maximum Slowdown Time	103.85	965.09	340.04	323.99	408.25
Effective Instances	407	436	478	493	499
Original Total Runtime	6784.8274	9480.7426	13166.2388	12340.3685	12614.6295
Simplified Total Runtime	1889.6190	7393.0903	12691.4633	11087.2024	13308.9542
Overall Speedup	3.59x	1.28x	1.04x	1.11x	0.95x
Original Conflicts	89317025	110993022	148917430	164029442	175687924
Simplified Conflicts	33294297	87352418	140771535	149084141	181429975
Conflict Reduction Rate	-62.72%	-21.30%	-5.47%	-9.11%	+3.27%

4.5. Performance Evaluation of Heuristic Guidance

In Section 4.4, we evaluated the performance of the hard-constraint variable fixing strategy. Although this approach can yield substantial speedups, it inevitably compromises the robustness of SAT solving by introducing the risk of incorrect unit clauses. To eliminate this risk while preserving logical soundness, we further evaluate a soft-constraint Guidance strategy, in which the model's predictions are used solely as branching heuristic hints for the Kissat solver. This strategy fundamentally differs from NeuroCore. NeuroCore repeatedly invokes a neural model during the solving process of a traditional CDCL solver to predict branching heuristics based on the evolving conflict state. Regardless of how the invocation schedule is optimized, frequent model calls during conflict propagation incur significant computational overhead. In contrast, our approach aims to invoke the trained model only once prior to solving, allowing variables that are predicted to have a strong influence on the solving process to be prioritized early in conflict propagation. This design reduces the overall number of conflicts and improves solver efficiency without introducing runtime inference overhead.

To incorporate high-confidence neural predictions as heuristic guidance while maintaining the logical completeness of SAT solving, we implement a non-intrusive phase initialization mechanism within the parsing stage of the Kissat solver. Specifically, we extend the standard DIMACS file parser. Although the DIMACS specification treats lines beginning with the character c as comments to be ignored by solvers, our modified parser intercepts comment lines containing a special GUIDANCE tag. When the parser encounters a line marked with c GUIDANCE,it reads the subsequent sequence of literals and maps their predicted polarities into Kissat's Saved Phases array. In this way, the solver's initial branching preferences are biased according to the neural network's predictions, while the underlying CDCL algorithm, conflict analysis, and backtracking mechanisms remain entirely unchanged. This design ensures that the guidance strategy is soundness-preserving, lightweight, and fully compatible with existing solver infrastructures. The corresponding pseudocode is shown below:

Table 10. Modifications to the src/parse module of Kissat to support heuristic guidance

Algorithm 1 Neural-Guided Phase Initialization in Kissat
Require: DIMACS file stream F, Solver instance S
1: Initialize: S.phases.saved ← zeros array
2: while not end of file F do
3: Read line L from F
4: if L starts with "p cnf" then
5: Parse N_vars, N_clauses
6: S.reserve_memory(N_vars) {Allocate memory for phases}
7: else if L starts with "c GUIDANCE" then
8: skip "c GUIDANCE" prefix
9: for each literal l in L do
10: if l == 0 then
11: break
12: end if
13: idx ← abs(l)
14: polarity ← sign(l)
15: {Inject Neural Hint into Saved Phases}
16: if idx ≤ S.N_vars then
17: S.phases.saved[idx] ← polarity
18: end if
19: end for
20: else
21: Parse standard DIMACS clauses or comments
22: end if
23: end while
24: return Initialized Solver S

The mathematical interpretation of this soft-constraint strategy is to initialize the preferred branching direction of the VSIDS decision heuristic. Let the neural network's prediction for variable xi be $\hat{y}_i \in \{0,1\}$. When the prediction confidence exceeds a predefined threshold, the internal phase state of the solver is initialized as

$$S_i \leftarrow \begin{cases} 1 & \text{if Network predicts } x_i = \text{True} \\ -1 & \text{if Network predicts } x_i = \text{False} \end{cases} \tag{14}$$

Unlike the hard-constraint strategy that adds unit clauses, this soft constraint only influences the solver's initial search trajectory. If the neural prediction leads to conflicts, Kissat's native CDCL mechanisms—including conflict analysis and non-chronological backtracking—will automatically override these initial phase assignments. Consequently, the solver's completeness is strictly preserved: if a satisfying assignment exists, the solver is guaranteed to find it.

Because the guidance acts as a soft constraint, incorrect predictions do not cause irrecoverable errors. For example, even if the model predicts x1=0.6 while the correct assignment is false, Kissat will naturally correct this choice during subsequent conflicts. Therefore, we evaluate a wide range of confidence thresholds, including 0.5, 0.6, 0.7, 0.8, and 0.95, to cover different confidence regimes.

For modern SAT solvers, random SAT instances with fewer than 200 variables (under the SR distribution) are considered trivial; Kissat typically solves them within two seconds or less. Due to the increased structural complexity of 3-SAT, comparative experiments for the guidance strategy are

conducted on datasets with 250–450 variables. Across all evaluated datasets, the results exhibit consistent behavior: regardless of the confidence threshold, the observed speedup remains stable within the range of 1.02×–1.11×, indicating that this strategy is largely insensitive to hyperparameter selection. This robustness arises from the solver's ability to automatically override incorrect phase assignments through conflict-driven learning.

Although the acceleration achieved by the Guidance strategy is smaller than that of the addVar hard-constraint approach, its zero-risk property makes it particularly suitable for scenarios where solution correctness is of paramount importance. Compared to the aggressive pruning behavior of addVar, Guidance functions more like providing the solver with a high-quality initial search direction rather than enforcing irreversible decisions.Table 11 reports the experimental results on the 250-variable dataset.

Table 11. Evaluation of the soft-constraint guidance strategy on the 250-variable dataset under different confidence thresholds compared with Kissat

Performance Metrics	Prediction Accuracy				
	>0.5	>0.6	>0.7	>0.8	>0.95
Total Instances	1000	1000	1000	1000	1000
Double Timeouts	0	0	0	0	0
Additional Solved Instances	0	0	0	0	0
Simplified Timeouts	0	0	0	0	0
Number of Slowdown Cases	397	350	347	344	295
Total Slowdown Time	390.81	285.79	325.97	321.07	225.41
Average Slowdown Time	0.98	0.82	0.94	0.93	0.76
Maximum Slowdown Time	11.08	11.79	10.75	9.51	7.16
Effective Instances	1000	1000	1000	1000	1000
Original Total Runtime	3966.0166	3966.0166	3966.0166	3966.0166	3966.0166
Simplified Total Runtime	3774.2981	3571.6187	3610.9450	3581.5359	3638.7113
Overall Speedup	1.05x	1.11x	1.10x	1.11x	1.09x
Original Conflicts	84158010	84158010	84158010	84158010	84158010
Simplified Conflicts	83611074	83326606	83297226	83948773	84776366
Conflict Reduction Rate	-0.65%	-0.99%	-1.02%	-0.25%	+0.73%

4.6. Model Effectiveness and Ablation Study

To further investigate the individual contributions of the proposed multi-slot global update mechanism (Global Slots) and the gated residual perceptron, we conduct a series of ablation experiments on the random 3-SAT dataset with 100 variables. This problem scale is selected because it offers a balanced trade-off between representational complexity and experimental efficiency, enabling frequent architectural modifications while preserving sufficient structural difficulty. Starting from a basic MPNN backbone, we construct four model variants to systematically evaluate the effectiveness of different components:

M0: The original NeuroSAT architecture, which relies solely on local message passing via an MPNN. Feature transformations are implemented using a standard MLP (Linear–ReLU–Linear). This model serves as the baseline for measuring the net performance gains introduced by subsequent enhancements.

M1: A variant of M0 in which the standard MLP is replaced by the proposed gated residual MLP. This variant is designed to assess the role of gating mechanisms in suppressing noisy messages, as well as the effect of residual connections in facilitating gradient flow and stabilizing training.

M2: A variant of M0 augmented with the multi-slot global Transformer module, while retaining the standard MLP for local updates. This configuration isolates the contribution of global latent slots in capturing long-range dependencies that are inaccessible to purely local message passing.

M3: The full model proposed in this work, which integrates both the gated residual MLP and the multi-slot global update mechanism.

As shown in Table 12, introducing the gated residual perceptron (M1) yields an accuracy improvement of 1.17%, indicating that the gating units effectively suppress the propagation of uninformative signals, while the LayerNorm and residual structure enhance training stability. The results of M2 further demonstrate that the primary bottleneck of conventional MPNNs lies in their locality bias. By incorporating global slots as information hubs, the model is able to directly capture logical constraints between variables that are distant in the graph, thereby correcting latent assignment errors that would otherwise persist. The complete model (M3) achieves the highest accuracy among all variants, confirming that the gated residual local updates and the multi-slot global mechanism are complementary and jointly essential for optimal performance.

Table 12. Ablation results analyzing the effects of global slots and gated residual MLP

ID	Core components	ValAcc (%)	improvements
M0	MPNN+VanillaMLP	78.58	-
M1	MPNN+GatedMLP	79.75	+1.17%
M2	MPNN+GlobalSlots	81.05	+2.47%
M3	MPNN+Slots+Gated	81.92	+3.34%

The global slots provide a broader and more holistic view of the problem structure, while the gated MLP ensures precise transmission and effective fusion of this global information with local representations. The ablation results confirm that both proposed components independently contribute to the performance improvements of the neural SAT solver, and that their combination yields the strongest predictive performance.

We further analyze the impact of the number of global slots. Increasing the slot count leads to a modest improvement in predictive accuracy; however, due to the additional computational overhead introduced by the Transformer module, the training cost increases nearly multiplicatively. As a result, the marginal performance gains are not commensurate with the increased training time. Based on empirical evaluation, we find that setting the number of slots to 8 achieves the best trade-off between accuracy and efficiency, and thus adopt this configuration in all experiments.

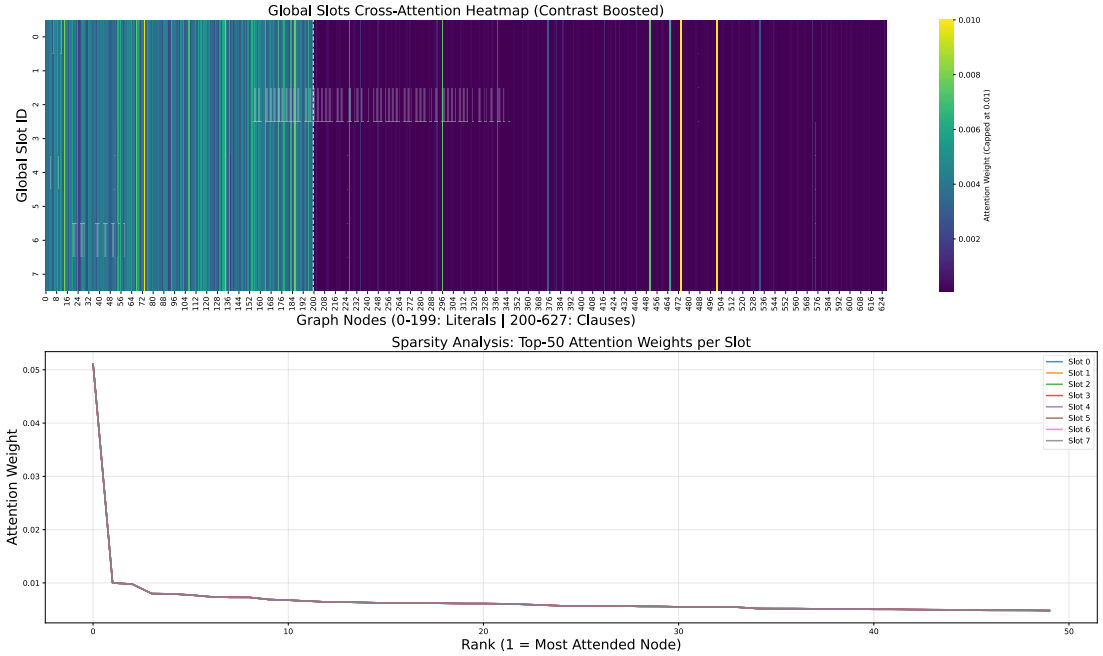


Figure 8. Cross-attention visualization of the multi-slot global Transformer on a 100-variable instance

As the multi-slot global update mechanism constitutes a core innovation of this work, we further validate its effectiveness through attention weight visualization on trained models using random 3-SAT instances with 100 variables. Specifically, we visualize the attention weights produced in the final iteration. To enhance visual contrast and avoid distortion caused by a small number of extreme values, the maximum attention weight is clipped to 0.01, preventing outliers from dominating the color scale. The resulting visualization is shown in Fig. 8. A notable observation is that, after multiple iterations, different global slots exhibit highly consistent attention distribution patterns. This phenomenon arises because the initially independent global latent vectors progressively exchange information through the self-attention mechanism during training, ultimately converging toward a coordinated global representation. From a spatial perspective, attention is relatively dense over the literal nodes, whereas it is sparse over clause nodes. This suggests that the model successfully identifies the core conflicting regions or unsatisfied clauses within the logical dependency chain. Interestingly, despite the overall sparsity in the clause region, the highest individual attention scores are often assigned to clause nodes, indicating that the model selectively highlights critical clauses while focusing on literals for detailed reasoning. This behavior implies that the model is capable of pinpointing logical bottlenecks and conflict centers when aggregating information from literals. Overall, the visualization analysis provides strong qualitative evidence that the proposed multi-slot global mechanism effectively filters redundant information and robustly captures the key long-range dependencies and logical bottlenecks that determine formula satisfiability.

5. CONCLUSION

This paper addresses the inherent limitations of existing message-passing neural network–based SAT solvers in modeling long-range dependencies, and proposes a hybrid neural architecture that integrates multi-slot global attention with gated message passing. By introducing a set of learnable global latent vectors that are independent of the underlying graph topology, the model establishes an explicit *global broadcast channel*, effectively mitigating information dissipation and over-smoothing issues commonly observed in deep graph networks. In addition, the proposed gated residual MLP significantly enhances the robustness of feature extraction by suppressing noisy messages during the propagation process.

Extensive experiments on large-scale random 3-SAT benchmarks validate the effectiveness of the proposed approach. Ablation studies demonstrate that the global slot mechanism and the gated components contribute independently and substantially at the architectural and feature-transformation levels, respectively. Furthermore, joint experiments with the state-of-the-art CDCL solver Kissat reveal two fundamentally different application paradigms and their trade-offs. The first is a hard-constraint strategy, where high-confidence variable assignments are directly fixed to aggressively reduce the search space. The second is a conservative guidance strategy, which injects neural predictions as branching heuristics, preserving solver completeness and achieving zero error rate, making it suitable for scenarios with strict correctness requirements.

Despite these advances, several limitations remain. The model is trained exclusively using supervision from satisfiable (SAT) instances, which limits its ability to explicitly reason about unsatisfiable (UNSAT) proofs. Moreover, as problem scale further increases (e.g., beyond 450 variables), confidence calibration under the hard-constraint strategy begins to degrade, indicating that the model's generalization capability over extremely deep logical chains can be further improved. Future work will explore unsupervised and semi-supervised learning paradigms, incorporating satisfiability-aware unsupervised loss functions to enable learning from UNSAT instances and unlabeled data. In addition, to achieve a better balance between solving efficiency and completeness, we plan to develop a dynamic decision mechanism that adaptively switches between hard constraints and soft guidance based on the current search state, which constitutes a key direction for future research.

REFERENCES

[1] Alyahya, Tasniem Nasser, Mohamed El Bachir Menai, and Hassan Mathkour. "On the structure of the boolean satisfiability problem: a survey." ACM Computing Surveys (CSUR) 55.3 (2022): 1-34.

[2] Cook, Stephen A. "The complexity of theorem-proving procedures." Logic, automata, and computational complexity: The works of Stephen A. Cook. 2023. 143-152.

[3] Vizel, Yakir, Georg Weissenbacher, and Sharad Malik. "Boolean satisfiability solvers and their applications in model checking." Proceedings of the IEEE 103.11 (2015): 2021-2035.

[4] Li, Min, et al. "On EDA-driven learning for SAT solving." 2023 60th ACM/IEEE Design Automation Conference (DAC). IEEE, 2023.

[5] Darwiche, Adnan. "New advances in compiling CNF to decomposable negation normal form." Proc. of ECAI. Citeseer, 2004.

[6] Marques-Silva, Joao, Inês Lynce, and Sharad Malik. "Conflict-driven clause learning SAT solvers." Handbook of satisfiability (2009): 131-153.

[7] Scarselli, Franco, et al. "The graph neural network model." IEEE transactions on neural networks 20.1 (2008): 61-80.

[8] Selsam, Daniel, et al. "Learning a SAT solver from single-bit supervision." arXiv preprint arXiv:1802.03685 (2018).

[9] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[10] Formisano, Andrea, and Flavio Vella. "On multiple learning schemata in conflict driven solvers." CEUR Workshop Proceedings Volume 1231, 2014, Pages 133-146. Vol. 1231. CEUR-WS, 2014.

[11] Biere, Armin, and Andreas Fröhlich. "Evaluating CDCL variable scoring schemes." International conference on theory and applications of satisfiability testing. Cham: Springer International Publishing, 2015.

[12] Cherif, Mohamed Sami, Djamal Habet, and Cyril Terrioux. "Kissat mab: Combining vsids and chb through multi-armed bandit." SAT COMPETITION 2021 (2021): 15.

[13] Sorensson, Niklas, and Niklas Een. "Minisat v1. 13-a sat solver with conflict-clause minimization." SAT 2005.53 (2005): 1-2.

[14] Guo, Wenxuan, et al. "Machine learning methods in solving the boolean satisfiability problem." Machine Intelligence Research 20.5 (2023): 640-655.

[15] Xu, Lin, et al. "SATzilla: portfolio-based algorithm selection for SAT." Journal of artificial intelligence research 32 (2008): 565-606.

[16] Devlin, David, and Barry O'Sullivan. "Satisfiability as a classification problem." Proc. of the 19th Irish Conf. on Artificial Intelligence and Cognitive Science. 2008.

[17] Selsam, Daniel, and Nikolaj Bjørner. "Guiding high-performance SAT solvers with unsat-core predictions." International conference on theory and applications of satisfiability testing. Cham: Springer International Publishing, 2019.

[18] Zhang, Wenjie, et al. "NLocalSAT: Boosting local search with solution prediction." arXiv preprint arXiv:2001.09398 (2020).

[19] Shi, Zhengyuan, et al. "Satformer: Transformer-based unsat core learning." 2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD). IEEE, 2023.

[20] Chang, Wenjing, Mengyu Guo, and Junwei Luo. "Predicting the satisfiability of Boolean formulas by incorporating gated recurrent unit (GRU) in the Transformer framework." PeerJ Computer Science 10 (2024): e2169.

[21] Cameron, Chris, et al. "Predicting propositional satisfiability via end-to-end learning." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 04. 2020.