Complementarity-Guided Neural Architecture Search for Encrypted Traffic Classification

Zhiguo Hu*, Yadong Lu*, Xi Ma†, Lifeng Guo*, Guoqing Liu*, and Kaikai Yang‡
*College of Computer and Information Technology, Shanxi University, Taiyuan, China
†Unit 96901 PLA, China
‡Shanxi University of Finance and Economics, Taiyuan, China
Email: {huzhiguotj, luyadong, lfguo}@sxu.edu.cn, sxxaepmx@126.com, guoqingl1001@163.com, ykkay@sxufe.edu.cn

***Abstract*—Deep learning-based encrypted traffic classification often relies on fixed network architectures with simple fusion strategies for multi-modal traffic features. As a result, feature complementarity and redundancy are not explicitly considered, which can limit model efficiency and generalization. Neural Architecture Search (NAS) provides a promising approach to automatic architecture optimization, but its application to large heterogeneous search spaces remains computationally expensive and slow to converge. In this work, we propose a complementarity-guided monte carlo tree search with evolutionary fine-tuning framework (CG-MCTS-NAS). The framework defines a structured search space over five heterogeneous traffic features, including packet length, inter-arrival time, raw bytes, byte entropy, and flow-level statistics. A complementarity matrix derived from feature correlations is incorporated into the PUCT strategy of MCTS, biasing the search toward more complementary feature combinations and reducing redundant exploration. Based on this design, a two-stage optimization strategy is employed, combining global architecture search with evolutionary fine-tuning of hyperparameters. Experiments on multiple real-world datasets show that CG-MCTS-NAS consistently outperforms recent state-of-the-art methods in classification performance, while reducing the required MCTS search iterations by approximately 55.6%. The complete code for the proposed method is available at https://github.com/yadonglu7-hue/CG-MCTS-NAS.**

***Index Terms*—Neural Network Applications, Monte Carlo Tree Search, Encrypted traffic classification, Efficient and Tiny Neural Networks.**

I. Introduction

With the rapid evolution of Internet applications and growing user demands for privacy and data security, encryption protocols like SSL/TLS have become the widely used in modern network communications, making most network service traffic "payload-invisible" [1]–[3]. Against this backdrop, accurate identification of encrypted traffic application types is critical to core tasks including network security protection, refined traffic management, quality of service (QoS) assurance, and abnormal behavior detection [4]–[6]. However, while encryption mechanisms effectively protect communication semantics, they obscure the discriminative information utilized by traditional port number or Deep Packet Inspection (DPI)-based methods [7]. Moreover, they reduce the behavioral distinguishability of different service flows, enabling malicious traffic to be more easily disguised in massive normal background traffic and thus greatly increasing the difficulty of encrypted traffic classification and security auditing [8].

Currently, encrypted traffic classification methods are mainly categorized into three types: traditional machine learning, deep learning, and NAS methods. Traditional machine learning methods model the statistical traffic features (e.g., packet length distribution, flow duration) primarily, with representative works including AppScanner [9] and CUMUL [10]. Such methods feature lightweight models and low computational overhead, yet their over-reliance on feature engineering limits generalization ability in complex network environments. In contrast, deep learning methods adopt an end-to-end learning mechanism to extract high-dimensional representations from raw bytes or packet sequences automatically, rendering them better suited to capturing complex nonlinear traffic patterns. Typical approaches encompass single-modal methods (e.g., FS-Net [11], ET-BERT [12]) and multi-feature fusion methods (e.g., BPF-DAG [13], FlowMiner [14]). Nevertheless, these methods predominantly rely on manually designed architectures, which require extensive expert knowledge and costly trial-and-error processes. Such paradigms struggle to efficiently explore optimal fusion structures within the vast search space of heterogeneous features. To mitigate these limitations, NAS has been introduced to automate model design (e.g., CIL-MTC [15]). However, existing NAS methods often fail to effectively distinguish core feature combinations from redundant structures, thereby hampering the balance between search efficiency and classification performance.

Motivations: Encrypted traffic exhibits multiple modal features (raw bytes, packet length, and inter-arrival time) with significant discrepancies in discriminative power and noise sensitivity, whose effective utilization relies on modeling feature complementarity and redundancy. Nevertheless, most existing methods adopt fixed architectures or simple fusion strategies, failing to capture the dynamic contributions of different modalities. Although NAS is introduced to realize automated optimization of classification models for addressing the above limitations, it still confronts key challenges such

This work was partially supported by the National Natural Science Foundation of China under Grants 62572290 and 62472267, the Shanxi Provincial Central Government Guided Local Science and Technology Development Fund Project under Grant YDZJSX2025B001, and the Key RD Program of Shanxi Province, China, under Grant 202202020101004. (Corresponding authors: Zhiguo Hu and Kaikai Yang; e-mail: huzhiguotj@sxu.edu.cn, ykkay@sxufe.edu.cn.)

as low search efficiency and invalid structure exploration in highly heterogeneous encrypted traffic scenarios.

Issue 1: Information Redundancy and Blind Combination in Heterogeneous Multi-Modal Feature Fusion

Issue Description: Encrypted traffic has distinct multi-modal heterogeneity, with features varying in the form and information density of discriminative cues. Most existing methods rely on simple concatenation or linear weighting, without explicitly modeling inter-modal complementarity and redundancy. Blind feature stacking increases computational overhead and degrades model generalization.

Issue 2: Efficiency Bottlenecks and Space Explosion of NAS in Heterogeneous Feature Spaces

Issue Description: Applying NAS (Neural Architecture Search) to encrypted traffic classification greatly expands the search space. Encompassing network operators, connection structures, modality selection and feature fusion, this leads to a sharp rise in search complexity. Without domain prior knowledge, general NAS strategies cannot distinguish high-value structures from redundant combinations, leading to search wandering in low-discriminative or highly correlated architectures and slow convergence.

To resolve these issues, this paper introduces CG-MCTS-NAS, a complementarity-guided automated architecture search framework. At its core, the framework incorporates feature complementarity as prior knowledge to bias the search toward high-information-gain feature combinations in the large heterogeneous search space. First, we build a five-dimensional hierarchical search space (covering packet length sequences, inter-arrival time, raw bytes, byte entropy, and flow-level statistics) to enable deep multi-modal feature fusion. Second, we introduce an information-theoretic complementarity matrix to refine MCTS (monte carlo tree search) heuristics, by quantifying inter-modal information gain, we efficiently prune redundant search branches. Finally, we adopt a two-stage optimization strategy consisting of macroscopic topology search and microscopic hyperparameter evolution to co-optimize multi-modal fusion architectures and model hyperparameters, yielding the optimal classification model under computational budget constraints.

The main contributions of this work are summarized as follows:

- We present CG-MCTS-NAS, an automated framework for encrypted traffic classification that jointly optimizes multi-modal fusion structures and network architectures via a two-stage search strategy.
- We develop a feature complementarity modeling approach that quantifies redundancy and incremental information among heterogeneous traffic views, providing an explicit prior for feature selection and fusion.
- We incorporate the complementarity prior into the PUCT formulation of MCTS, which reduces the required MCTS search iterations by approximately 55.6% while maintaining strong classification performance.

II. Related Work

Existing encrypted traffic classification methods mainly fall into three categories: traditional machine learning [9], [10], [16]–[19], deep learning [11], [14], [20]–[26], and NAS [13], [27]–[32]. Among these, the latter two exhibit stronger modeling and generalization capabilities in scenarios involving complex encryption and traffic obfuscation.

Deep learning-based methods: These methods can be further divided into two main types based on the feature modeling approach: single-modal and multi-modal methods. The first type focuses on mining local spatial or temporal patterns in traffic data. For example, Wang et al. proposed a 1D-CNN [20] to extract spatial patterns from raw byte streams, while Liu et al. introduced FS-Net [11], which uses a bidirectional recurrent network to capture temporal dependencies in packet length sequences. Lin et al. proposed TSCRNN [21], a hybrid spatiotemporal architecture that combines CNNs and LSTMs to improve classification accuracy. However, these methods model single-dimensional features with fixed architectures, which limits their discriminative power and results in sub-optimal performance in complex scenarios.

The second type includes multi-modal fusion methods, which integrate different traffic features to improve model robustness and semantic understanding. Lin et al. proposed ET-BERT [12], using a Transformer for self-supervised pre-training on large-scale unlabeled traffic, while Niu et al. introduced FlowMiner [14], which employs GNNs (graph neural networks) to capture inter-flow correlations. Liu et al. proposed YaTC [22], a masked autoencoder-based method that learns robust representations via reconstruction. While these methods improve performance, they are computationally expensive due to the high overhead of pre-training, and many of them still rely on simple fusion strategies. Moreover, these approaches often do not explicitly model inter-view complementarity and redundancy, which can cause feature interference.

NAS-based methods: Neural Architecture Search (NAS) aims to automatically discover the optimal network architecture within a predefined search space. Recent studies have shown that NAS can significantly improve traffic recognition performance. For instance, Malekghaini et al. proposed an automated architecture search method for lightweight networks [29], while Zhang et al. employed a proximal iteration strategy to balance performance and efficiency in IoT applications [30]. Wang et al. integrated multi-head self-attention with partial architecture search to improve detection accuracy in industrial applications [31]. Xie et al. demonstrated that NAS can generalize well across different classification tasks [13]. Zhao et al. proposed CIL-MTC, which introduces a scalable architecture search mechanism to address topology adaptation for malicious traffic [15].

These works have made significant progress in automating the design of neural networks, breaking through the performance limitations of handcrafted architectures. However, when applied to encrypted traffic classification, NAS still faces two major challenges. First, most existing methods lack explicit

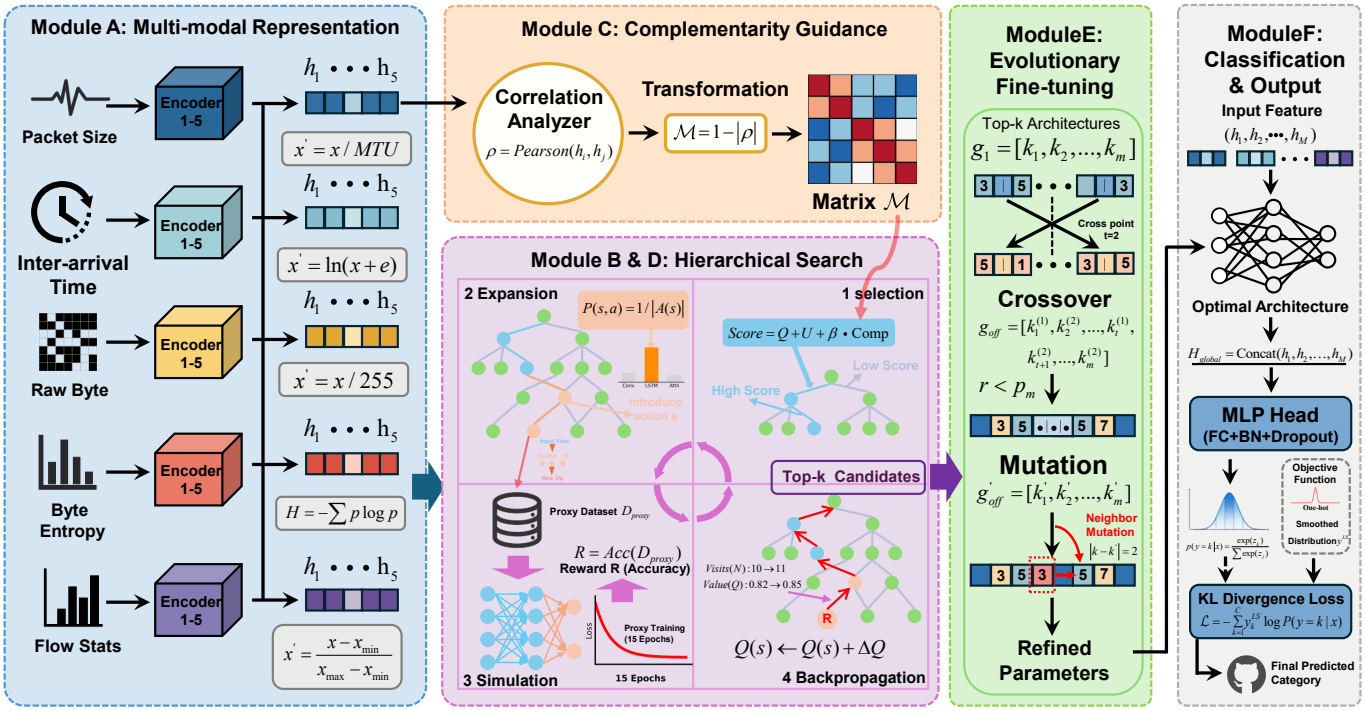

Fig. 1: Overview of CG-MCTS-NAS. The framework integrates multi-modal feature fusion with a two-level hybrid search strategy for encrypted traffic classification. First, Modules A and C extract five heterogeneous traffic features and construct a complementarity prior matrix $\mathcal{M}$. Second, within the hierarchical search space of Module B, the CG-MCTS algorithm (Module D) performs global topology search and pruning. Third, Module E applies an evolutionary strategy to fine-tune local architectural hyperparameters (e.g., kernel sizes). Finally, Module F fuses the selected multi-modal features and produces the classification output with label smoothing.

modeling of feature complementarity and redundancy, leading to the blind combination of features and resulting in redundant or conflicting structures. Second, NAS methods applied to large-scale heterogeneous search spaces suffer from low efficiency and slow convergence, as the search process often explores redundant combinations, especially in the absence of domain-specific priors.

III. Methodology

We propose a complementarity-guided monte carlo tree search with evolutionary fine-tuning framework (CG-MCTS-NAS), which is designed to automatically discover the optimal neural network architecture with both high discriminability and modal complementarity in the encrypted traffic scenario. The overall framework is illustrated in Fig. 1.

A. Multi-Modal Feature Extraction

To address the limitations of single features in identifying complex encrypted traffic, this module extracts five types of heterogeneous views in parallel from a multi-dimensional space to construct a comprehensive feature representation tensor. The multi-modal raw feature space is defined as V, where V_{sub} denotes its subset, as shown in Eq. (1):

$$V = \{v_{\text{len}}, v_{\text{iat}}, v_{\text{byte}}, v_{\text{ent}}, v_{\text{stat}}\}, \quad V_{\text{sub}} \subseteq V. \tag{1}$$

Packet length features reflecting application-layer interaction patterns are normalized by the maximum transmission unit (MTU) as $x' = x/\text{MTU}$ to yield the feature vector v_{len}; inter-arrival time characterizing transmission rhythm is processed via logarithmic transformation with a smoothing factor $e = 10^{-6}$ ($x' = \ln(x+e)$) to generate v_{iat}; raw byte features carrying protocol fingerprints are mapped from ASCII codes (0–255) to the range $[0,1]$ ($x' = x/255$) to produce v_{byte}; byte entropy measuring payload randomness is computed using the Shannon entropy formula $H = -\sum p \log p$ (where p denotes the occurrence probability of byte values) to obtain v_{ent}; and flow statistical features summarizing macroscopic traffic properties are standardized via Min-Max normalization $\left(x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}\right)$ to generate v_{stat}.

B. Hierarchical Search Space

Given the heterogeneous nature of multi-modal inputs, instead of searching arbitrary neural network topologies, this paper focuses on discovering effective multi-view modeling strategies, and constructs a hierarchical search space S, which is formed by the Cartesian product of three subspaces: view selection, operator mapping, and fusion strategy, as shown in Eq. (2). Here, S_{V}, S_{O}, and S_{F} denote the view selection, operator mapping, and fusion strategy subspaces, respectively:

$$S = S_{\text{V}} \times S_{\text{O}} \times S_{\text{F}}. \tag{2}$$

Each element in S specifies a candidate architecture by defining its input views, encoders, and fusion strategy.

First, we determine the active input views via view selection S_V: the initial view set is V, and a binary selector vector $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5]^T \in \{0,1\}^5$ is introduced. Here, $\sigma_i = 1$ indicates that view v_i is selected, while $\sigma_i = 0$ indicates exclusion. The constraint $\sum_{i=1}^{5} \sigma_i \geq 1$ is imposed to avoid an empty set, and the final selected view subset $V_{\text{sub}} = \{v_i \mid \sigma_i = 1,\ i \in \{1,2,3,4,5\}\}$ is obtained, which determines the number and composition of input branches in the network.

Next, we assign view-specific feature encoders via operator mapping S_{O}: we define an operator pool $O = O_{\text{seq}} \cup O_{\text{spa}} \cup O_{\text{stat}}$, where $O_{\text{seq}} = \{\text{1D-CNN, LSTM, Dilated Conv}\}$ (sequence processing operators), $O_{\text{spa}} = \{\text{ResNet-18}\}$ (spatial processing operators), and $O_{\text{stat}} = \{\text{MLP}\}$ (statistical processing operators). To avoid invalid or inefficient architectures, operator assignments are constrained by view-operator compatibility. We define a mapping function $\phi : V_{\text{sub}} \to O$, which must satisfy the view-operator compatibility constraint as shown in Eq. (3), thereby assigning an appropriate feature extraction operator to each selected view,

$$\phi(v_i) = \begin{cases} o_{\text{seq}} \in O_{\text{seq}} & v_i \in \{v_{\text{len}}, v_{\text{iat}}, v_{\text{ent}}\}, \\ o_{\text{spa}} = \text{ResNet-18} & v_i = v_{\text{byte}}, \\ o_{\text{stat}} = \text{MLP} & v_i = v_{\text{stat}}. \end{cases} \tag{3}$$

Finally, we execute the fusion strategy S_{F}: we define a fusion strategy set $F = \{\text{Concatenation, Weighted Sum, Multi-Head Attention}\}$, and introduce an aggregation function $\Gamma : \{h_i\}_{v_i \in V_{\text{sub}}} \to H_{\text{agg}}$. The corresponding strategy expressions are as follows: Concatenation: $\Gamma_{\text{concat}}(\{h_i\}) = \text{Concat}(h_1, h_2, \ldots, h_k)$, where k is the number of selected views; Weighted Sum: $\Gamma_{\text{weight}}(\{h_i\}) = \sum_{i=1}^{k} \alpha_i h_i$, where α_i are learnable weights satisfying $\sum_{i=1}^{k} \alpha_i = 1$; Multi-Head Attention: $\Gamma_{\text{attn}}(h_i) = \text{MultiHead}(Q, K, V)$, where Q, K, and V are obtained by linear transformation of $\{h_i\}$;

This step enables effective multi-view feature aggregation, providing a unified representation for subsequent classification and evaluation.

C. Complementarity Matrix Construction

In the vast search space, blindly combining views with extremely high correlation (e.g., v_{ent} and v_{stat} are highly correlated in certain scenarios) leads to model redundancy. To address this, we introduce a complementarity degree matrix M as a guidance mechanism for the search.

For any view v_i ($i \in \{1,2,3,4,5\}$), corresponding to packet length, inter-arrival time, raw byte, byte entropy, and flow statistics, the pre-trained encoder processes it to output a fixed-dimension feature vector $E(v_i) \in \mathbb{R}^d$, where d denotes the feature dimension and $E(\cdot)$ represents the pre-trained encoder.

The linear correlation between views v_i and v_j is measured by the Pearson correlation coefficient $\rho(E(v_i), E(v_j))$, whose specific formula is given in Eq. (4):

$$\rho(E(v_i), E(v_j)) = \frac{Cov(E(v_i), E(v_j))}{\sigma_{E(v_i)} \cdot \sigma_{E(v_j)}}. \tag{4}$$

Here, the covariance is defined as $Cov(E(v_i), E(v_j)) = \mathbb{Z}[(E(v_i) - \mu_{E(v_i)}) \odot (E(v_j) - \mu_{E(v_j)})]$, where $\mathbb{Z}$ denotes the expectation operator, $\mu_{E(v_i)}$ is the mean of $E(v_i)$, and $\odot$ represents the element-wise product. The standard deviation is $\sigma_{E(v_i)} = \sqrt{\mathbb{Z}[(E(v_i) - \mu_{E(v_i)})^2]}$, and $\sigma_{E(v_j)}$ is defined analogously.

The complementarity between views v_i and v_j, denoted as $C(v_i, v_j)$, is defined as the degree of non-correlation between their feature vectors, as given in Eq. (5). Here, $C(v_i, v_j) = 1$ indicates that the two views are completely complementary (no redundancy), while $C(v_i, v_j) = 0$ indicates that the two views are completely redundant,

$$C(v_i, v_j) = 1 - |\rho(E(v_i), E(v_j))|. \tag{5}$$

Although Pearson correlation primarily captures linear dependencies between features, in this work we employ it solely as a lightweight and stable complementarity guidance signal to provide an effective prior bias in the large-scale search space, rather than to precisely model inter-view statistical dependencies. As an approximate measure, Pearson correlation offers clear advantages in terms of computational efficiency and numerical stability, which are sufficient to support complementarity ranking and screening during the architecture search stage. Based on this definition, we construct a 5×5 complementarity matrix $M \in \mathbb{R}^{5\times5}$, where each element represents the degree of complementarity between a pair of views, i.e., $M = [M_{ij}]_{5\times5}$ and $M_{ij} = C(v_i, v_j)$.

D. Complementarity-Guided MCTS

In the **selection phase** of monte carlo tree search (MCTS), to achieve more efficient path screening in the complex search space, we improved the evaluation criterion for action selection. As shown in Eq. (6), when the system decides the next action a (i.e., selecting a new feature or operator), it not only refers to its historical performance and exploration count, but also explicitly introduces a complementarity reward term $Comp(a, s)$:

$$a_t = \arg\max_a \left(Q(s,a) + C_p \cdot P(s,a) \frac{\sqrt{N(s)}}{1 + N(s,a)} + \beta \cdot \text{Comp}(a,s) \right). \tag{6}$$

Here, $Q(s,a)$ denotes the average historical accuracy of action a (exploitation), C_p is the exploration constant, and $P(s,a)$ is the prior probability of selecting action a under state s. To avoid double-counting complementarity guidance, we adopt a uniform prior $P(s,a) = 1/|A(s)|$, where $A(s)$ is the set of candidate actions available at state s. Moreover, $N(s)$ denotes the visit count of state s, and $N(s,a)$ denotes the number of times action a has been selected at state s. The complementarity reward term is defined as $\text{Comp}(a,s) = \frac{1}{|s|} \sum_{x \in s} \mathcal{M}_{a,x}$, which measures the average complementarity between the candidate action a and the already selected set s according to the complementarity matrix $\mathcal{M}$, where $|s|$ is the number of selected elements and $\mathcal{M}_{a,x}$ denotes the complementarity score between a and x. When $|s| = 0$, we

define $\text{Comp}(a, s) = 0$. Note that $\text{Comp}(a, s)$ is only applied to view-selection actions; for operator-level actions, it is set to zero. Finally, β is a balance factor controlling the strength of complementarity guidance.

Expansion: If the current node is an expandable node, a new action a_{new} (e.g., adding an LSTM layer or an input view) is expanded according to the search space S, and its prior probability $P(s, a)$ is initialized.

Simulation: Starting from the newly expanded node, lightweight training is performed for 15 epochs on the dataset subset D_{proxy}. The proxy accuracy is calculated as Eq. (7),

$$Acc_{proxy} = \frac{1}{|D_{proxy}|} \sum_{(x,y)\in D_{proxy}} I\big(\hat{y}(x) = y\big), \tag{7}$$

where $\hat{y}(x)$ is the predicted class of sample x by the current architecture, and $I(\cdot)$ is an indicator function (taking 1 for correct predictions and 0 otherwise). This accuracy serves as the evaluation reward, i.e., $R = Acc_{proxy}$.

Backup: The reward R is propagated backward along the selected path. For each visited edge (s, a), the visit count is first incremented, and the corresponding action value is then updated using incremental averaging:

$$\begin{aligned} N(s, a) &\leftarrow N(s, a) + 1, \\ Q(s, a) &\leftarrow Q(s, a) + \frac{R - Q(s, a)}{N(s, a)}. \end{aligned} \tag{8}$$

Here, $N(s, a)$ denotes the number of times action a has been selected at state s, and $Q(s, a)$ represents the estimated value of taking action a at state s. This update rule maintains $Q(s, a)$ as the running average of observed rewards, ensuring a stable and unbiased value estimation during the search.

E. Evolutionary Fine-tuning

While MCTS excels in macroscopic architecture exploration, it is less efficient for microscopic hyperparameter fine-tuning (e.g., kernel size and stride). Consequently, we introduce a post-convergence evolutionary phase to optimize these micro-level architectural hyperparameters via genetic operations.

Each architecture's micro-parameter set is defined as a hyperparameter encoding vector $\boldsymbol{g} = [k_1, k_2, \ldots, k_m]$, where the dimension matches the number of operator layers m. Here, k_i denotes the convolution kernel size of the i-th operator, satisfying the constraint $k_i \in K,\ K = \{3, 5, 7, 9\}$.

Crossover: Fusion of superior hyperparameter segments. Two parent genes $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ are randomly selected from the Top-k superior architectures output by MCTS, and an offspring gene $\boldsymbol{g}_{off}$ is generated via segment exchange. First, a crossover point $t \in \{1, 2, \ldots, m-1\}$ is randomly chosen (splitting the gene into two parts), then the latter segments of the parents are swapped. The offspring gene is generated as Eq. (9), where $k_i^{(1)}$ is the i-th element of g_1, and $k_i^{(2)}$ is that of g_2,

$$\boldsymbol{g}_{off} = \big[k_1^{(1)}, \ldots, k_t^{(1)}, k_{t+1}^{(2)}, \ldots, k_m^{(2)}\big]. \tag{9}$$

Mutation: Neighborhood exploration of hyperparameters. For the offspring gene $\boldsymbol{g}_{off}$ generated by crossover, each element k_i is independently mutated with a probability p_m via neighborhood replacement. Mutation is triggered if a randomly sampled value $r \sim \text{Uniform}(0, 1)$ satisfies $r < p_m$.

The neighborhood mapping for elements in K is defined as $\mathbb{N}(k_i) = \{k \in K \mid |k - k_i| = 2\}$. The mutated value is $k_i' = \text{RandomChoice}(\mathbb{N}(k_i))$, and the final mutated gene vector is given in Eq. (10):

$$\boldsymbol{g}'_{off} = \big[k_1', k_2', \ldots, k_m'\big]. \tag{10}$$

F. Classification Head & Training Objective

After completing the two-stage hybrid architecture search combining monte carlo tree search (MCTS) and evolutionary fine-tuning, the system employs the resulting optimal architecture for final classification, where multi-modal features are fused to produce the recognition result.

Following the architecture search, the selected fusion strategy is instantiated, and a concatenation operator is used to merge the view feature vectors $\{h_1, h_2, \ldots, h_M\}$ into a global feature vector H_{global}, as shown in Eq. (11), where M is the number of views selected in the view selection phase ($M \in \{1, 2, 3, 4, 5\}$, determined by the complementarity-guided search strategy), and h_i is the feature vector of the i-th view. Subsequently, H_{global} is fed into an MLP classification head consisting of 2 fully connected (FC) layers, interleaved with batch normalization (BN) layers and Dropout layers to suppress overfitting and improve generalization ability:

$$H_{global} = \text{Concat}(h_1, h_2, \ldots, h_M). \tag{11}$$

Next, the Softmax function is used to calculate the prediction probability for each class, as shown in Eq. (12), where z_k is the output of the k-th neuron in the classification head, and C is the total number of classes,

$$P(y = k \mid x) = \frac{\exp(z_k)}{\sum_{j=1}^{C} \exp(z_j)}. \tag{12}$$

Finally, the system introduces a Label Smoothing mechanism to smooth the distribution, as shown in Eq. (13), where ϵ is the smoothing factor and $\mathbb{I}(\cdot)$ is an indicator function (taking 1 when $k = y$ and 0 otherwise),

$$y_k^{LS} = (1 - \epsilon) \cdot \mathbb{I}(k = y) + \frac{\epsilon}{C}. \tag{13}$$

The final training objective is to minimize the cross-entropy loss between the predicted distribution P and the smoothed distribution y^{LS}, as shown in Eq. (14),

$$\mathcal{L} = -\sum_{k=1}^{C} y_k^{LS} \log P(y = k \mid x). \tag{14}$$

IV. Experiments

Datasets We use the USTC-TFC2016 [33] and ISCX VPN-nonVPN [34] datasets to construct four evaluation tasks, as shown in Table I. For each dataset, samples are randomly split into training, validation, and test sets with a ratio of 8:1:1. The training set is used for model training and architecture search, the validation set for hyperparameter selection, and the test set is reserved exclusively for final performance evaluation. All statistics and prior information used during architecture search are computed solely based on the training split.

Data Preprocessing We apply standard preprocessing to ensure input consistency, including label encoding, mean imputation for missing values, and Min–Max normalization to the $[0, 1]$ range. In addition, categories with extremely few samples are removed to improve training stability.

TABLE I: Summary of Dataset Composition and Tasks.

Dataset	Task ID	Task Type	Objective
USTC-TFC2016	Task 1	Malicious Detection	Benign vs. Malicious
	Task 2	Family Identification	20-class Fine-grained
ISCX-VPN-2016	Task 3	Service ID	5-class VPN
ISCX-NonVPN	Task 4	Service ID	5-class Regular

Implementation Details First, during the MCTS search phase, we set the exploration constant $C_p = 0.1$ and guidance factor $\beta = 0.8$, and quickly locked the backbone architecture within 50 iterations with a depth limit of 6. Subsequently, using an evolutionary algorithm with a mutation probability of 0.1, we performed 30 generations of fine-grained optimization on hyperparameters such as convolution kernel size, hidden layer dimension, and dropout rate. For model training, we adopted the Adam optimizer with a cosine annealing strategy and weighted cross-entropy loss, and configured optimal batch sizes for the search phase (Batch Size = 64) and the final training phase (Batch Size = 128), respectively. All experiments were conducted on a workstation equipped with an AMD Ryzen 5 7500F CPU (6 cores), 32 GB DDR5 memory, and an NVIDIA GeForce RTX 5060 Ti GPU (8 GB).

TABLE II: Ablation Study on Model Components.

Model Variants	AC	PR	RC	F1
w/o CG	0.9210	0.9180	0.9190	0.9185
w/o Evolution	0.9500	0.9480	0.9490	0.9485
Random Search	0.8780	0.8730	0.8750	0.8740
w/o NAS (Fixed ResNet-18)	0.9150	0.9120	0.9130	0.9125
w/o NAS (Fixed AttnLSTM)	0.9600	0.9500	0.9620	0.9585
w/ Average Fusion	0.9320	0.9534	0.9391	0.9457
w/ Concatenation Fusion	0.9420	0.9400	0.9410	0.9405
Ours (Default)	**0.9733**	**0.9733**	**0.9741**	**0.9733**

Evaluation Metrics Performance is evaluated using Accuracy (AC), Precision (PR), Recall (RC), and F1-score (F1).

Baselines To evaluate the performance of the framework, we compare five recent and representative baseline models with publicly available implementations. Specifically, 1D-CNN (ICON, 2017) [20] is based on local sequential features; FlowMiner (INFOCOM, 2025) [14] and Opendect (TIFS, 2025) [35] exploit heterogeneous representation and association mining; ET-BERT (WWW, 2022) [12] and YATC (AAAI, 2023) [22] leverage pre-trained large model representations.

A. Main Results

The main results are reported in Table III. On USTC-task1, our method (AC=0.9994, F1=0.9993) exceeds the second-best model Opendect by 0.66 pp in AC and 0.65 pp in F1. On USTC-task2, our method (AC=0.9583, F1=0.9582) improves over Opendect by 1.31 pp in AC and 1.30 pp in F1. On ISCX-VPN, our method (AC=0.9733, F1=0.9733) outperforms ET-BERT by 0.34 pp in AC and 0.80 pp in F1. On ISCX-nonVPN, our method (AC=0.9660, F1=0.9657) surpasses FlowMiner by 5.54 pp in AC and 5.58 pp in F1. We attribute these gains to the two-stage search strategy: complementarity-guided MCTS favors low-redundancy feature combinations, while evolutionary fine-tuning refines key architectural hyperparameters.

B. Ablation Study

As shown in Table II, replacing CG-MCTS-guided search with random search results in a 9.93% F1-score drop. Removing complementarity guidance (w/o CG) leads to a 5.48% performance degradation compared to the default model. Replacing NAS with manually designed architectures (ResNet-18 or AttnLSTM) causes performance drops ranging from 1.48% to 6.08%, clearly indicating the necessity of automated architecture and fusion design. In addition, removing evolutionary fine-tuning results in a 2.48% performance decrease. These results indicate that CG-MCTS-guided search and NAS are critical for adapting to encrypted traffic scenarios, whereas evolutionary fine-tuning provides additional but relatively smaller performance gains.

C. Efficiency Verification

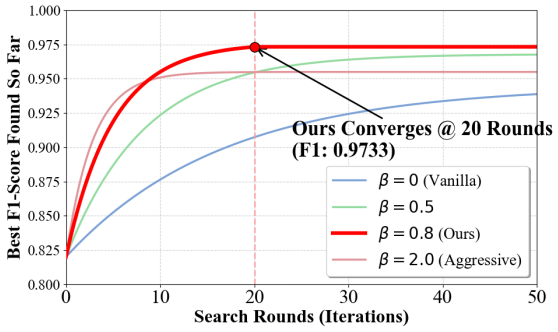

Fig. 2: Optimal F1-Score Conv. Curves Comparison Under Different Complementarity Factors β.

To evaluate the search efficiency of the Complementarity-guided (CG) mechanism under limited computational budgets, we vary the guidance factor β on the ISCX-VPN dataset and

TABLE III: Performance Comparison on Multiple Datasets.

Model	USTC-task1				USTC-task2				ISCX-VPN				ISCX-nonVPN			
	AC	RC	PR	F1	AC	RC	PR	F1	AC	RC	PR	F1	AC	RC	PR	F1
1D-CNN [20]	0.9500	0.9500	0.9250	0.9333	0.8543	0.8542	0.8554	0.8544	0.9540	0.9313	0.9472	0.9387	0.8483	0.7910	0.8264	0.8042
ET-BERT [12]	0.9913	0.9922	0.9912	0.9916	0.9335	0.9305	0.9315	0.9310	0.9700	0.9631	0.9676	0.9653	0.8407	0.8450	0.8480	0.8430
YATC [22]	0.9600	0.9556	0.9460	0.9508	0.9100	0.9153	0.8964	0.9058	0.9600	0.9564	0.9584	0.9574	0.9100	0.8934	0.9040	0.8987
FlowMiner [14]	0.9828	0.9831	0.9828	0.9827	0.9410	0.9438	0.9410	0.9387	0.9511	0.9516	0.9511	0.9512	0.9106	0.9106	0.9103	0.9099
Opendect [35]	0.9928	0.9929	0.9928	0.9928	0.9452	0.9449	0.9451	0.9452	0.9433	0.9424	0.9432	0.9429	0.9028	0.9028	0.9050	0.9028
Ours	**0.9994**	**0.9990**	**0.9996**	**0.9993**	**0.9583**	**0.9593**	**0.9583**	**0.9582**	**0.9733**	**0.9741**	**0.9733**	**0.9733**	**0.9660**	**0.9658**	**0.9660**	**0.9657**

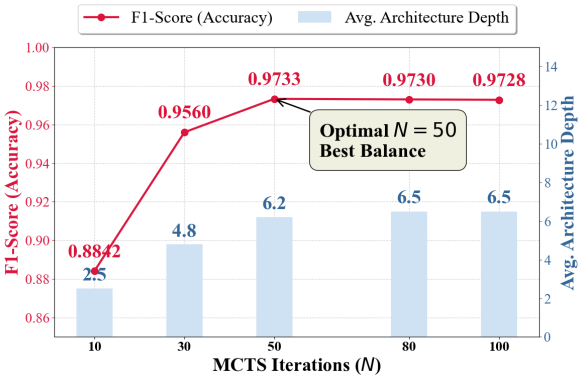

Fig. 3: The Impact of MCTS Iterations N on Architecture Depth and Classification Performance.

TABLE IV: Impact of Complementarity Factor β on Search Performance and Efficiency.

Complementarity β	Convergence Rounds	Efficiency Gain	F1-Score
$\beta = 0$ (vanilla)	45	Baseline	0.9185
$\beta = 0.5$	22	+51.1%	0.9642
$\beta = 0.8$	20	+55.6%	0.9733
$\beta = 2.0$	12	+73.3%	0.9512

analyze the convergence behavior. As shown in Table IV, vanilla MCTS ($\beta = 0$) requires 45 iterations to approach the performance upper bound, whereas incorporating complementarity guidance significantly accelerates convergence. In particular, setting $\beta = 0.8$ reduces the required iterations to 20, corresponding to a 55.6% reduction in convergence rounds.

Fig. 2 further shows that the F1-score initially increases and then decreases with larger β, reaching the optimum value of 0.9733 at $\beta = 0.8$. Overly strong guidance (e.g., $\beta = 2.0$) leads to premature convergence by overemphasizing feature orthogonality, thereby degrading classification performance. Accordingly, $\beta = 0.8$ is adopted as a balanced choice between search efficiency and representation effectiveness.

All architecture search experiments were conducted on a CPU-based platform without GPU acceleration. The proposed CG-MCTS converged within a limited wall-clock time using multi-core CPUs, making GPU-days inapplicable.

D. Key Hyperparameter Sensitivity Analysis

(1) Stability Test of MCTS Iteration Count N

Since the iteration count N governs the depth and breadth of exploration in the hierarchical search space, we evaluate performance across $N \in [10, 100]$ to identify the optimal configuration. As shown in Fig. 3, increasing N exhibits phased effects: when N increases from 10 to 50, the average architecture depth rises from 2.5 to 6.2 and the F1-score improves to 0.9733. Beyond $N = 50$, the F1-score plateaus, slightly decreasing to 0.9728 at $N = 100$, while the average depth stabilizes near 6.5, indicating diminishing returns. Therefore, $N = 50$ is adopted as the default configuration.

(2) Sensitivity of Fine-Tuning Hyperparameters

We further examined the impact of evolutionary population size on the performance of Module E. As shown in Fig. 4, increasing the population size from 10 to 20 boosts the F1 score significantly to 0.9733, stemming from the enhanced genetic diversity of a larger population that strengthens the algorithm's global search capability to avoid local optima. However, beyond 20, the F1 score plateaus with negligible accuracy gains: the MCTS stage predefines the macro architecture, leaving little room for microscopic optimization—further increasing the size would drastically elevate computational costs for marginal gains. Thus, a population size of 20 is chosen as the default configuration for microscopic optimization after comprehensive trade-offs.

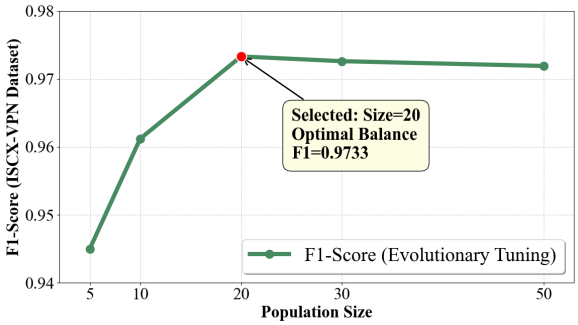

Fig. 4: Impact of population size on F1-score performance.

V. Conclusion

This paper presents CG-MCTS-NAS, a Complementarity-Guided monte carlo tree search framework with evolutionary

fine-tuning, to address feature redundancy, manual architecture design, and low search efficiency in encrypted traffic classification. By introducing a complementarity matrix as prior guidance, the proposed framework efficiently prunes the hierarchical search space and enables effective multi-modal feature fusion via a two-stage optimization strategy. Experiments on real-world datasets demonstrate that CG-MCTS-NAS consistently outperforms existing methods in terms of classification performance and generalization, highlighting its effectiveness for automated architecture search in encrypted traffic analysis. Future work will explore extensions to more complex scenarios and practical deployment.

REFERENCES

[1] E. Rescorla, "The transport layer security (tls) protocol version 1.3," Tech. Rep., 2018.

[2] J. Iyengar, M. Thomson *et al.*, "Quic: A udp-based multiplexed and secure transport," in *RFC 9000*, 2021.

[3] S. Farrell, "Ech interoperability report, version 1," *DEfO Project Report, Tech. Rep., Jan*, 2025.

[4] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[5] Q. Zhou, L. Wang, H. Zhu, T. Lu, and V. S. Sheng, "Wf-transformer: Learning temporal features for accurate anonymous traffic identification by using transformer networks," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 30–43, 2023.

[6] T. Van Ede, R. Bortolameotti, A. Continella, J. Ren, D. J. Dubois, M. Lindorfer, D. Choffnes, M. Van Steen, and A. Peter, "Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic," in *Network and distributed system security symposium (NDSS)*, vol. 27, 2020.

[7] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[8] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer networks*, vol. 31, no. 23-24, pp. 2435–2463, 1999.

[9] V. F. Taylor, R. Spolaor, M. Conti, and I. Martinovic, "Appscanner: Automatic fingerprinting of smartphone apps from encrypted network traffic," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016, pp. 439–454.

[10] A. Panchenko, F. Lanze, J. Pennekamp, T. Engel, A. Zinnen, M. Henze, and K. Wehrle, "Website fingerprinting at internet scale." in *NDSS*, vol. 1, 2016, p. 23477.

[11] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "Fs-net: A flow sequence network for encrypted traffic classification," in *IEEE INFOCOM 2019-IEEE Conference On Computer Communications*. IEEE, 2019, pp. 1171–1179.

[12] X. Lin, G. Xiong, G. Gou, Z. Li, J. Shi, and J. Yu, "Et-bert: A contextualized datagram representation with pre-training transformers for encrypted traffic classification," in *Proceedings of the ACM Web Conference 2022*, 2022, pp. 633–642.

[13] Y. Shi, G. Li, J. Wu, J. Li, and H. Fang, "Bpf-dag: Byte-packet-flow features fusion via dynamic attributed graph for reliable encrypted traffic classification," *IEEE Transactions on Information Forensics and Security*, vol. 21, pp. 197–211, 2025.

[14] H. Xu, C. Si, S. Li, Z. Cheng, C. Wang, J. Xie, P. Sun, and Q. Liu, "Flowminer: A powerful model based on flow correlation mining for encrypted traffic classification," in *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*. IEEE, 2025, pp. 1–10.

[15] X. Zhang, Y. Wang, T. Ohtsuki, G. Gui, C. Yuen, M. Di Renzo, and H. Sari, "Malware traffic classification via expandable class incremental learning with architecture search," *IEEE Transactions on Information Forensics and Security*, 2025.

[16] A. S. Jacobs, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, and L. Z. Granville, "Ai/ml for network security: The emperor has no clothes," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1537–1551.

[17] J. Gu and S. Lu, "An effective intrusion detection approach using svm with naïve bayes feature embedding," *Computers & Security*, vol. 103, p. 102158, 2021.

[18] S.-J. Xu, G.-G. Geng, X.-B. Jin, D.-J. Liu, and J. Weng, "Seeing traffic paths: Encrypted traffic classification with path signature features," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 2166–2181, 2022.

[19] S. Ramraj and G. Usha, "Unsupervised feature learning methodology for tree based classifier and svm to classify encrypted traffic," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 2, 2023.

[20] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE international conference on intelligence and security informatics (ISI)*. IEEE, 2017, pp. 43–48.

[21] K. Lin, X. Xu, and H. Gao, "Tscrnn: A novel classification scheme of encrypted traffic based on flow spatiotemporal features for efficient management of iiot," *Computer Networks*, vol. 190, p. 107974, 2021.

[22] R. Zhao, M. Zhan, X. Deng, Y. Wang, Y. Wang, G. Gui, and Z. Xue, "Yet another traffic classifier: A masked autoencoder based traffic transformer with multi-level flow representation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 4, 2023, pp. 5420–5427.

[23] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for internet of things," *IEEE access*, vol. 5, pp. 18 042–18 050, 2017.

[24] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, "Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE access*, vol. 6, pp. 1792–1806, 2017.

[25] P. Sirinam, M. Imani, M. Juarez, and M. Wright, "Deep fingerprinting: Undermining website fingerprinting defenses with deep learning," in *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, 2018, pp. 1928–1943.

[26] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescapé, "Xai meets mobile traffic classification: Understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[27] X. He, K. Zhao, and X. Chu, "Automl: A survey of the state-of-the-art," *Knowledge-based systems*, vol. 212, p. 106622, 2021.

[28] R. Lyu, M. He, Y. Zhang, L. Jin, and X. Wang, "Network intrusion detection based on an efficient neural architecture search," *Symmetry*, vol. 13, no. 08, p. 1453, 2021.

[29] N. Malekghaini, E. Akbari, M. A. Salahuddin, N. Limam, R. Boutaba, B. Mathieu, S. Moteau, and S. Tuffin, "Automl4etc: Automated neural architecture search for real-world encrypted traffic classification," *IEEE Transactions on Network and Service Management*, vol. 21, no. 3, pp. 2715–2730, 2023.

[30] X. Zhang, L. Hao, G. Gui, Y. Wang, B. Adebisi, and H. Sari, "An automatic and efficient malware traffic classification method for secure internet of things," *IEEE Internet of Things Journal*, vol. 11, no. 5, pp. 8448–8458, 2023.

[31] Y. Wang and X. Xu, "An industrial internet of things-oriented malicious traffic detection network with neural network partial architecture search," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*. IEEE, 2023, pp. 1820–1825.

[32] S. Dong, Y. He, Z. Zhou, H. Luo, X. Wei, A. C. Kot, and Y. Gong, "Class-independent increment: An efficient approach for multi-label class-incremental learning," *arXiv preprint arXiv:2503.00515*, 2025.

[33] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International conference on information networking (ICOIN)*. IEEE, 2017, pp. 712–717.

[34] G. D. Gil, A. H. Lashkari, M. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related features," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP 2016)*. SciTePress Setúbal, Portugal, 2016, pp. 407–414.

[35] Q. Meng, J. Tao, Q. Yuan, G. Li, Y. Wang, B. Gao, and S. Lu, "Detection of unknown attacks through encrypted traffic: A gaussian prototype-aided variational autoencoder framework," *IEEE Transactions on Information Forensics and Security*, 2025.