

Genetic-Algorithm Feature Selection for Evaluating Self-Supervised Transformers and Gradient-Boosted Trees in Flow-Based Intrusion Detection

Mohammad F. Al-Hammouri^{1,*}, Bandi Vamsi², Muder M. AlMi'ani³, Elvira Al Bataineh⁴, and Ali Al Bataineh^{5,*}

¹ Department of Computer Engineering, Faculty of Engineering, The Hashemite University, Zarqa 13133, Jordan.

² Department of Artificial Intelligence, Madanapalle Institute of Technology & Science (MITS), Deemed to be University, Madanapalle 517326, Andhra Pradesh, India.

³ Gulf university for science and technology, MIS department, Kuwait.

⁴ College of Graduate and Continuing Studies, Norwich University, Northfield, VT, USA.

⁵ Computer Information Science, Higher Colleges of Technology, Abu Dhabi, United Arab Emirates.

* Corresponding authors: alhammouri@hu.edu.jo and aalbataineh@hct.ac.ae

Received: 22 June 2026 **Accepted:** 05 August 2026 **Published:** 10 October 2026

DOI: 10.15849/ijasca.166 **License:** Open access under CC BY 4.0. Authors retain copyright.

Publisher: Al-Zaytoonah University of Jordan (ZUJ), Amman, Jordan.

Cite this article: M. F. Al-Hammouri, B. Vamsi, M. M. AlMi'ani, E. Al Bataineh, and A. Al Bataineh, "Genetic-algorithm feature selection for evaluating self-supervised transformers and gradient-boosted trees in flow-based intrusion detection," *Int. J. Advance Soft Compu. Appl.*, vol. 18, no. 3, pp. 296–316, 2026, doi: <https://doi.org/10.15849/ijasca.166>.

ABSTRACT

This study examines whether a self-supervised Transformer improves flow-based network intrusion detection when every model receives the same features, selected by a genetic algorithm (GA) whose logistic-regression surrogate is independent of the evaluated models. Transformer architectures and self-supervised pretraining are increasingly proposed for intrusion detection, but their advantage over strong classical baselines on tabular flow data is rarely tested under controlled conditions. We introduce a model-agnostic genetic-algorithm feature-selection stage and use it to compare a self-supervised tabular Transformer (SST-IDS) with Random Forest, XGBoost, CNN, LSTM, and TabTransformer on CIC-IDS-2017 and UNSW-NB15. The protocol uses identical seed-specific feature sets and stratified splits for every paired model, five random seeds on both datasets, full 20-epoch pretraining and 20-epoch fine-tuning schedules with early stopping, and paired significance tests. XGBoost achieved 0.9456 ± 0.0067 accuracy and 0.9856 ± 0.0078 AUC on UNSW-NB15, compared with 0.9257 ± 0.0040 and 0.9781 ± 0.0076 for SST-IDS. On the disclosed 20,000-flow CIC sample, XGBoost achieved 0.9963 ± 0.0007 accuracy and 0.9998 ± 0.0001 AUC. Full training increased SST-IDS mean UNSW-NB15 AUC from 0.9705 to 0.9781 relative to the reduced schedule, but did not reverse the comparative result. Zero-shot transfer was unstable: SST-IDS reached mean AUC 0.6126 ± 0.1960 , while Random Forest and XGBoost remained below 0.5 on average. SHAP explanations and identically measured deployment costs complete the analysis.

Keywords: intrusion detection; genetic algorithm; feature selection; self-supervised learning; Transformers; gradient-boosted trees.

1 Introduction

Network intrusion detection systems (IDS) are a core component of modern security operations [1], [2], and machine learning has become the dominant approach to building them [13]. In recent years, the field has followed the broader trajectory of deep learning: attention-based architectures and self-supervised pretraining, which transformed natural language processing and computer vision, are increasingly proposed for network traffic analysis. A growing research has explored the use of Transformer-based models for intrusion detection, with many studies reporting strong performance and suggesting that these models outperform traditional machine learning approaches.

These claims, however, deserve careful examination for two main reasons. First, methodological limitations are common in the literature. Many studies optimize the proposed model extensively while evaluating baseline methods with default settings, rely on a single train–test split of one dataset, or omit statistical significance testing. As a result, the reported performance gains may not hold under a more rigorous and controlled evaluation. The second is domain-specific: network-flow records are tabular data, and a substantial body of evidence outside the security literature indicates that well-configured

gradient-boosted tree ensembles remain highly competitive, and frequently superior, to deep architectures on tabular problems [34], [35]. If that pattern holds for flow-based intrusion detection, then practitioners choosing between a Random Forest and a Transformer-based detector need to know, because the two options differ sharply in engineering complexity, inference cost, and operational maturity.

This paper therefore asks a deliberately simple question and answers it under controlled, fully reproducible conditions: does a self-supervised Transformer earn its place in a flow-based IDS pipeline when all models use features selected by a neutral evolutionary search? We study a representative self-supervised tabular Transformer configuration, which we refer to as SST-IDS, that treats each network-flow feature as a token, pretrains a multi-head self-attention encoder with a masked-feature reconstruction objective on unlabeled flows, and then fine-tunes the encoder for binary benign/malicious classification. A seeded genetic algorithm (GA) selects features using validation performance from a class-balanced logistic-regression surrogate and a parsimony penalty, so feature selection is not derived from either tree baseline. We evaluate SST-IDS against Random Forest, XGBoost, CNN, LSTM, and TabTransformer on CIC-IDS-2017 and UNSW-NB15. The study is organized around three research questions:

1. RQ1 (Detection performance). Under identical splits, identical feature sets, repeated random seeds, and paired statistical testing, how does the self-supervised Transformer compare with strong tree ensembles and with comparable neural baselines?
2. RQ2 (Cross-dataset transfer). How much detection ability survives a strictly zero-shot transfer from CIC-IDS-2017 to UNSW-NB15, two datasets collected in different environments, when the transfer representation is restricted to behavioral (non-identity) features?
3. RQ3 (Explainability and deployment cost). What features drive the model's decisions, globally and for individual errors, and what do the candidate models cost to operate in terms of per-flow latency and, for SST-IDS, parameter memory?

Our findings are direct. On RQ1, XGBoost is the strongest detector on both benchmarks, while adequate-budget SST-IDS remains below it on accuracy, F1, and AUC. The budget ablation confirms that longer training improves SST-IDS AUC but does not reverse the ranking. On RQ2, zero-shot CIC-to-UNSW transfer is weak and unstable for every model, with SST-IDS reaching mean AUC 0.6126 ± 0.1960 and the tree baselines remaining below 0.5 on average. On RQ3, SHAP attributions identify behavioral flow statistics as the dominant evidence, and identically measured amortized latency is 0.0403 ms for SST-IDS, 0.0758 ms for Random Forest, and 0.0015 ms for XGBoost.

We regard the negative component of these results as a contribution rather than a shortcoming. Positive results for attention-based IDS are abundant in the literature; carefully controlled evidence about when they do not outperform much simpler models is scarce, and it is precisely what a practitioner needs for model selection. The contributions of this study are:

1. A seeded binary genetic algorithm that selects model-agnostic feature subsets using a class-balanced logistic-regression surrogate, validation F1, and an explicit parsimony penalty.
2. A unified and fully reproducible evaluation protocol for flow-based IDS: identical stratified train/validation/test splits and identical selected feature sets for every paired model, five random seeds on both datasets, and paired significance tests reporting significant and non-significant comparisons.
3. Controlled evidence that strong gradient-boosted tree ensembles match or exceed a self-supervised Transformer on two heterogeneous flow benchmarks, a finding that the Transformer-based IDS literature rarely reports, together with an analysis of which differences are statistically significant.
4. A budget ablation that directly tests whether the Transformer result changes under the full 20+20 schedule, plus a GA-versus-RF-SHAP ablation that quantifies the effect of the feature-selection protocol.

5. A strictly zero-shot cross-dataset comparison of SST-IDS, Random Forest, and XGBoost using only semantically aligned behavioral features.
6. An explainability and deployment-cost analysis combining global SHAP attributions, local explanations for representative false-positive and false-negative decisions, and per-flow latency and model-size measurements.
7. A practical treatment of the documented labeling and feature-computation defects of CIC-IDS-2017 within the evaluation protocol, with mitigations and their limits made explicit.

The remainder of the paper is organized as follows. Section 2 reviews related work. Section 3 describes the study design: datasets, preprocessing, feature selection, the model under study, the baselines and fairness protocol, and the measurement procedures. Section 4 reports the results organized by research question. Section 5 discusses implications, recommendations for practitioners, and limitations. Section 6 concludes.

2 Related Work

Intrusion detection research has progressed from rule-based and signature-based methods to machine learning, deep learning, Transformer-based models, self-supervised learning, and explainable AI. Earlier IDS approaches were useful for detecting known attacks, but they often struggled with evolving attack behavior, high-dimensional traffic data, and previously unseen network patterns. Recent studies have used deep learning and attention-based architectures to improve detection performance; however, several practical challenges remain, including dependence on labeled data, limited transferability across datasets, high computational cost, and weak interpretability. This section reviews related work in five areas: traditional and optimized IDS methods, machine learning and deep learning for IDS, Transformer-based intrusion and anomaly detection, self-supervised learning in security, and explainable AI for IDS.

2.1 Traditional and Optimized IDS Methods

Traditional IDS relied on rule-based reasoning, statistical modeling, signature matching, and conventional machine learning methods to detect malicious activity in network traffic. These systems provided an important foundation for automated threat detection, but they often required manually designed rules, fixed thresholds, or hand-crafted features. As attack behavior changes, such systems may need frequent updates and may not adapt well to new or unknown attacks. Their limited adaptability has encouraged researchers to explore optimization-based, hybrid, and learning-based IDS models.

Megantara and Ahmad [7] proposed a hybrid machine learning method to improve network intrusion detection performance. Their work showed that combining learning strategies can improve detection accuracy compared with simple standalone classifiers. However, such methods still depend heavily on the quality of selected features and the availability of labeled training data. Naseri and Gharehchopogh [14] investigated feature selection using a farmland fertility algorithm to improve IDS performance. Their results confirmed that selecting informative features can improve classification effectiveness and reduce computational cost. However, feature-selection-driven IDS approaches often remain supervised and do not directly learn representations from unlabeled traffic.

Alotaibi et al. [16] proposed GWQBBA, a hybrid feature selection model that combines Grey Wolf Optimization and Quantum Binary Bat Algorithm for intrusion detection. The method was evaluated on the UNSW-NB15 dataset and used a Random Forest classifier to test the selected features. The reported results showed the value of optimized feature selection for IDS classification. However, the framework remains supervised and does not directly address self-supervised representation learning, Transformer-based feature modeling, or transfer behavior across heterogeneous network datasets.

Reed et al. [17] developed SA-IDS, a single-attribute intrusion detection method for Slow DoS attacks in IoT networks. The model uses a delta-time network attribute and a dynamic thresholding strategy to distinguish malicious slow nodes from benign traffic. This approach is lightweight and suitable for constrained IoT environments. However, it focuses on a narrow attack scenario and does not address

broader IDS settings involving multiple attack types, high-dimensional network-flow features, or explainable model outputs. Barik and Misra [18] introduced IDS-Anta, an open-source IDS tool designed to detect adversarial attacks [5] using methods such as ZOO and FGSM. Although the tool improves adversarial robustness and reproducibility, its main focus is adversarial defense rather than self-supervised learning, cross-dataset transfer assessment, or feature-level explanation for IDS decisions.

2.2 Machine Learning and Deep Learning for IDS

Machine learning and deep learning have improved IDS performance by enabling models to learn patterns directly from network traffic data. Classical machine learning methods, such as decision trees, support vector machines, Random Forest, Naive Bayes, and ensemble classifiers, have been widely used for binary and multi-class intrusion detection. Deep learning models, including CNNs, LSTMs, and hybrid architectures, can capture more complex traffic patterns and often outperform traditional models [3]. However, these approaches frequently depend on large labeled datasets, careful preprocessing, feature selection, and extensive tuning. Their decision-making process can also be difficult to interpret.

Thaljaoui [6] proposed an optimized deep CNN-LSTM model for IDS using UNSW-NB15. The model combined convolutional and recurrent components to improve the detection performance in network traffic data. Similarly, Udurume et al. [10] compared deep CNN-BiLSTM and machine learning methods for IDS, showing that hybrid deep learning models can improve classification performance. However, CNN-LSTM-based models can be computationally expensive and may not transfer consistently across traffic environments without careful tuning.

Najafi Mohsenabad and Tut [8] compared bio-inspired optimization algorithms for cybersecurity attack detection using the CSE-CIC-IDS 2018 dataset. Their findings emphasized the importance of optimization and feature selection for IDS performance. Umar et al. [20] examined the effects of feature selection and normalization on network intrusion detection. Their study showed that preprocessing and feature selection choices can significantly influence IDS performance. Zouhri et al. [11] likewise evaluated the impact of filter-based feature selection methods on intrusion detection system performance. Santos et al. [12] also evaluated the impact of preprocessing techniques on IDS performance, confirming that data preparation can strongly affect model accuracy and reliability. These studies support the need for clear preprocessing, feature selection, and data handling procedures when evaluating IDS models.

2.3 Transformer-Based Intrusion and Anomaly Detection

Transformer models have become increasingly common in anomaly detection and intrusion detection because self-attention can model relationships among input features without relying only on local or sequential patterns. Compared with CNNs and RNNs, Transformers can capture broader dependencies and adapt to structured or high-dimensional network-flow data. However, Transformer-based IDS models still face challenges related to computational cost, data imbalance, feature-token design, explainability, and cross-dataset validation.

Zeng et al. [15] proposed an adversarial Transformer architecture for multivariate time-series anomaly detection in IoT environments. The model uses Transformer encoders and adversarial learning to detect deviations from normal behavior. Although the method showed strong anomaly detection performance, it focused mainly on multivariate time-series anomaly detection rather than binary IDS classification using tabular network-flow features and SHAP-based feature attribution.

S. Ullah et al. [22] introduced TNN-IDS, a Transformer neural network-based intrusion detection system for MQTT-enabled IoT networks. The model uses multi-head attention to capture relationships among MQTT traffic attributes and detect malicious behavior. Although TNN-IDS showed strong performance in MQTT-based IoT traffic, its evaluation was mainly protocol-specific, leaving broader heterogeneous dataset transfer as an open issue. AL-Nomasy et al. [21] proposed a Transformer-based knowledge distillation model for explainable IDS. Their teacher-student design reduced model

complexity while maintaining detection performance. However, they focused on lightweight IDS and knowledge distillation rather than masked-feature self-supervised pretraining.

Rai et al. [25] proposed RTAD-HIS, a regulated Transformer architecture for anomaly detection in healthcare IoT systems. Their method modifies self-attention to improve anomaly detection and reduce false positives. The study demonstrated the usefulness of Transformer-based modeling in a healthcare IoT context, but broader validation across heterogeneous network datasets remains a challenge. Natsos and Symeonidis [19] proposed a Transformer-based malware detection approach using process resource utilization metrics. Although this work is not a network-flow IDS framework, it further supports the growing role of Transformer models in security-related detection tasks.

2.4 Self-Supervised Learning in Security

Self-supervised learning (SSL) has become an important direction in cybersecurity because it allows models to learn useful representations from unlabeled data. In intrusion detection, SSL methods commonly use pretext tasks such as reconstruction, masked prediction, contrastive learning, or temporal ordering. These methods can reduce dependence on labeled data and may improve adaptation to changing traffic patterns. However, pretext-task design, handling tabular network-flow features, and integrating explainability remain open research issues.

Radhakrishnan et al. [4] explored unsupervised representation learning for intrusion detection in Industrial Internet of Things environments. Their work showed the value of representation learning when labeled data is limited. Sarantos et al. [23] proposed a semi-supervised learning approach for IDS, using unlabeled data to improve detection under limited labeling conditions. These studies demonstrate the usefulness of learning from unlabeled or partially labeled traffic, but they do not fully resolve the need for interpretable, Transformer-based feature representation in tabular IDS settings.

Li et al. [26] proposed a Multi-View Graph Contrastive Representation Learning framework for intrusion detection in electric vehicle charging stations. The model constructs multiple graph views and applies contrastive learning to improve representation quality. The method achieved strong performance in its target domain, but scalability to larger and more heterogeneous network environments remains an open challenge. Hossain et al. [27] proposed a privacy-preserving self-supervised IDS for 5G-V2X networks by combining SSL with federated learning. Their results showed that SSL can improve performance when labeled data is limited, although robustness against advanced adversarial attacks requires further study.

2.5 Explainable AI for IDS

Although deep learning models can improve intrusion detection accuracy, they often operate as black boxes. This creates a challenge in cybersecurity, where analysts need to understand why traffic was classified as benign or malicious. Explainable AI methods, including SHAP, local interpretable model-agnostic explanations (LIME), fuzzy prototypes, attention visualization, and domain-knowledge-driven explanations, have been used to improve transparency in IDS models. These methods can help identify influential features and support analyst trust, but they also introduce challenges such as computational overhead, explanation stability, and the need for local explanations of correct and incorrect predictions.

Abou El Houda et al. [9] proposed an explainable deep learning framework for IoT-based IDS. Their work highlighted the importance of explanation in security applications where trust and transparency matter. Kalutharage et al. [24] introduced a neurosymbolic and domain-knowledge-driven explainable AI approach for IoT network attack detection and response. Their work showed that combining domain knowledge with learning models can improve interpretability and decision support.

Gu et al. [28] proposed S3PAFES, a semi-supervised soft prototype-based autonomous fuzzy ensemble system for network intrusion detection from data streams. The model uses collaborative pseudo-labeling, active learning, dynamic pruning, and sampling strategies to address class imbalance. Soft prototypes are used to support explainability and maintain robustness over time. However, challenges remain in concept drift handling, human intervention, and large-scale deployment. Prasad et al. [29]

developed an explainable AI framework for threat detection in SDN-based multicast systems using SHAP and LIME. The study emphasized both global and local explanations, but also noted computational overhead and scalability concerns.

F. Ullah et al. [30] introduced IDS-INT, a Transformer transfer learning method for imbalanced network traffic. The framework combines Transformer-based feature extraction, imbalance handling, and deep classification, and uses SHAP and LIME to interpret feature importance. This work is closely related to the present study because it combines Transformer-based learning with explainability. However, the present work focuses on masked feature prediction for self-supervised pretraining, binary IDS classification using selected network-flow features, and a controlled zero-shot transfer comparison across heterogeneous IDS datasets.

2.6 Positioning of This Study

The literature surveyed above shows a clear trajectory toward attention-based architectures, self-supervised objectives, and explainability tooling for intrusion detection, with individual papers typically proposing a new model and reporting favourable comparisons. Two gaps motivate the present study. First, controlled negative evidence is scarce: comparisons in which the baselines receive the same feature sets, splits, and tuning discipline as the proposed model, with repeated runs and statistical testing, are the exception rather than the rule, and studies reporting that simpler models match or beat the proposed architecture are rarer still. Second, the strength of gradient-boosted trees on tabular data, documented in benchmark studies outside the security literature [34], [35], should be tested directly in evaluations of Transformer-based IDS, because network-flow records are tabular. This study addresses both gaps: it subjects a representative self-supervised tabular Transformer (Section 3.5) and five widely used baselines to a single, fully specified protocol on two heterogeneous benchmarks, reports all outcomes including unfavourable ones, and releases the complete implementation. Relative to the two most closely related Transformer-based systems, TNN-IDS [22] and IDS-INT [30], the model studied here differs in that its Transformer is pretrained with a masked-feature self-supervised objective before any labels are used, and the evaluation differs in that it is designed to test, rather than to showcase, the architecture: identical conditions for all models, paired statistical tests, a behavioral-features-only transfer measurement, and local explanations for individual errors.

3 Study Design

This section specifies the complete experimental design. The guiding principle is that every model sees exactly the same data under exactly the same conditions, and every design choice is either fixed in the released configuration or reported here. Figure 1 summarizes the unified evaluation pipeline.

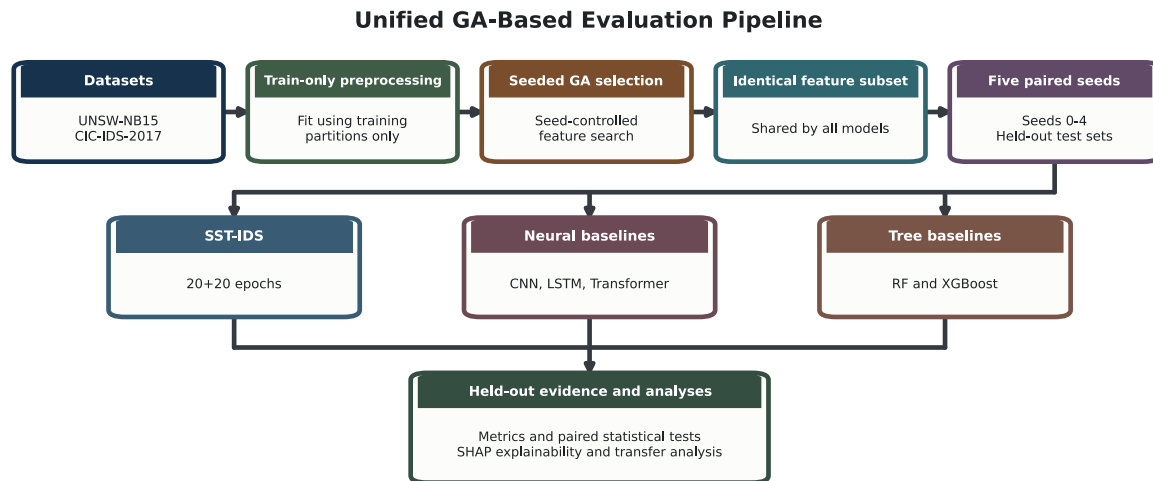


Figure 1. Unified GA-based evaluation pipeline. Train-only preprocessing and seeded feature selection produce an identical subset for SST-IDS and all five baselines; five paired seeds feed held-out metrics, statistical tests, explainability, and transfer analyses.

Table 1. Summary of the IDS datasets used in this study (Source cols., columns in the released file, including the label column and, for UNSW-NB15, the record identifier and attack-category columns; Samples, records before cleaning; FTP, File Transfer Protocol; SSH, Secure Shell; DoS, denial of service; DDoS, distributed denial of service).

Dataset	Subset / Traffic Scenario	Source cols.	Samples
UNSW-NB15 [31]	Training subset	45	175,341
UNSW-NB15 [31]	Testing subset (transfer target only)	45	82,332
CIC-IDS-2017 [32]	Monday – benign baseline	79	529,918
CIC-IDS-2017 [32]	Tuesday – FTP/SSH Brute Force	79	445,909
CIC-IDS-2017 [32]	Wednesday – DoS scenarios	79	692,703
CIC-IDS-2017 [32]	Thursday AM – Web Attack	79	170,366
CIC-IDS-2017 [32]	Thursday PM – Infiltration	79	288,602
CIC-IDS-2017 [32]	Friday AM – baseline traffic	79	191,033
CIC-IDS-2017 [32]	Friday PM – PortScan	79	286,467
CIC-IDS-2017 [32]	Friday PM – DDoS	79	225,745

3.1 Dataset Description

This study uses two publicly available, heterogeneous intrusion detection datasets, CIC-IDS-2017 and UNSW-NB15, which differ substantially in network environment, traffic characteristics, feature definitions, and attack types. Evaluating the framework on these two benchmarks tests it under genuinely different traffic conditions rather than on a single benchmark. Table 1 summarizes the datasets.

3.1.1 UNSW-NB15 Dataset

The UNSW-NB15 dataset was created by the Cyber Range Lab of the Australian Centre for Cyber Security [31]. It contains benign and malicious network traffic generated using modern attack scenarios. The dataset includes several attack categories, such as Generic, Exploits, DoS, Fuzzers, Worms, Reconnaissance, and Shellcode attacks. In this study, we used the training subset containing 175,341 records. The dataset includes numerical and categorical traffic features, such as packet counts, byte counts, flow duration, service type, protocol, and state information.

For binary intrusion detection, we mapped all attack categories to the attack class, and normal traffic to the benign class. The UNSW-NB15 training subset was used for within-dataset evaluation; the official UNSW-NB15 testing split served as the external evaluation set in the strictly zero-shot cross-dataset transfer comparison (Section 3.10).

3.1.2 CIC-IDS-2017 Dataset

The Canadian Institute for Cybersecurity created the CIC-IDS-2017 dataset [32]. It contains realistic benign and attack traffic collected over five working days in a simulated enterprise network environment. The dataset includes several attack scenarios, such as Brute Force, DoS/DDoS, PortScan, Infiltration, and Web attacks. The traffic was processed using CICFlowMeter, producing flow-based features such as flow duration, packet counts, byte counts, flow rates, packet length statistics, and inter-arrival time features.

In this study, the CIC-IDS-2017 files were analyzed to preserve day-wise traffic characteristics and attack distributions. The Monday file mainly represents benign baseline traffic. The Tuesday file includes benign traffic mixed with Brute Force activity. The Wednesday file contains denial-of-service scenarios. The Thursday files include Web Attack and Infiltration traffic. The Friday files contain three distinct traffic settings: Friday-Morning, Friday-Afternoon-PortScan, and Friday-Afternoon-DDoS. These files capture normal traffic, reconnaissance-oriented PortScan activity, and high-volume DDoS traffic, respectively. The flow CSVs used here (the “MachineLearningCVE” version) contain only behavioral flow statistics and a destination-port field; they do not include source/destination IP addresses or a protocol-identifier column, so the model cannot key on network identity.

We note that CIC-IDS-2017 has well-documented labeling and feature-computation issues. Independent audits have reported mislabeled flows, errors and inconsistencies in the original CICFlowMeter feature extraction, and duplicate or ambiguous records [33]. These defects can inflate reported performance if left unaddressed, which is consistent with the near-perfect scores obtained by the tree-based baselines on this dataset in our experiments (Table 7). To reduce, though not eliminate, the impact of these issues, we (i) remove records with missing or infinite feature values during preprocessing, (ii) pool the day-wise captures so that both classes are present and metrics are well defined, and (iii) report repeated runs and statistical tests rather than single-split point estimates. We nevertheless interpret the CIC-IDS-2017 results with caution and recommend that future work validate the framework on a corrected version of the dataset.

Analyzing the files separately allows the proposed framework to be evaluated under different traffic conditions within the same benchmark. This file-level analysis also shows how the model behaves across different attack types and traffic distributions before the external transfer assessment.

3.2 Data Preprocessing

We applied preprocessing to both datasets before model training. This step cleaned the raw traffic data, created a consistent binary classification format, and prepared the features for self-supervised and supervised learning. Table 2 summarizes the preprocessing operations applied across the datasets.

First, we identified missing values, infinite values, and invalid entries. Rows containing invalid values were removed or corrected using statistically appropriate replacement values when applicable. All numerical columns were converted into suitable numerical data types to ensure compatibility with the learning pipeline.

Second, for datasets that contain categorical fields, those features were encoded. In UNSW-NB15, the categorical fields protocol, service, and state were encoded, whereas the CIC-IDS-2017 flow features are entirely numeric and required no categorical encoding. Low-cardinality categorical variables were one-hot encoded when interpretability and dimensionality allowed. Higher-cardinality categorical variables were encoded using label encoding to avoid excessive feature expansion.

Third, numerical features were standardized using z-score normalization:

$$z = \frac{x - \mu}{\sigma} \quad (1)$$

where x is the original feature value, μ is the feature mean, and σ is the feature standard deviation. Standardization was important because network-flow features such as flow duration, byte counts, packet length statistics, and packet rates can have very different scales.

Finally, labels were converted into a binary format. Benign traffic was assigned label 0, and all attack classes were assigned label 1. For CIC-IDS-2017 and UNSW-NB15, multiple attack categories were grouped into a single attack class to support binary benign/malicious classification. This binary formulation is deliberate: it matches the alert/no-alert decision made at deployment time and keeps the paired six-model, five-seed protocol tractable; multi-class attack categorization is outside the scope of this study.

3.3 Dataset Sampling

UNSW-NB15 was retained without row reduction. The CIC files were sampled deterministically within capture and class using sampling seed 991, then pooled to 20,000 flows (Table 3). We retained this sample size due to computational resource constraints: training the four neural models over five seeds on a larger sample was not feasible within the available compute budget. The trim applies only to CIC sample size. Both datasets use the full 20+20 neural schedule and five experimental seeds. Consequently, the adequate-budget answer to RQ1 rests primarily on the complete UNSW-NB15 experiment, while the CIC provides corroborating evidence under the prior sample size.

Table 2. Preprocessing Operations Applied Across the IDS Datasets

Dataset Name	Features / Fields	Preprocessing Technique
CIC-IDS-2017 [32]	Flow Duration, Packet Length, Flow Packets/s	Z-score normalization
CIC-IDS-2017 [32]	Label	Binary encoding: Benign = 0, Attack = 1
CIC-IDS-2017 [32]	Destination Port, TCP flag counts	Retained as numeric features (the CIC-IDS-2017 flow CSVs contain no protocol/service categorical columns, so no label/one-hot encoding is applied)
CIC-IDS-2017 [32]	All numeric features	Removal of rows with NaN or infinite values
CIC-IDS-2017 [32]	All features	Type conversion to float32 or int64
UNSW-NB15 [31]	Protocol, Service, State	Label encoding or one-hot encoding
UNSW-NB15 [31]	sbytes, dbytes, spkts, dpkts	Z-score normalization
UNSW-NB15 [31]	Label	Binary encoding: Benign = 0, Attack = 1
UNSW-NB15 [31]	All features	Removal of rows with NaN or infinite values

Table 3. Dataset Size Before and After Preprocessing and Sampling

Dataset	Subset / Traffic Scenario	Source Rows	Final Rows	Sampling / Cleaning Method
UNSW-NB15 [31]	Official training subset	175,341	175,341	No row reduction
CIC-IDS-2017 [32]	Eight pooled capture files	2,830,743	20,000	Fixed-seed within-capture/class sampling, then pooling

3.4 Feature Selection and Feature Engineering

Feature selection uses a binary genetic algorithm. A chromosome $z \in \{0,1\}^d$ identifies the retained features. For each chromosome, a class-balanced logistic-regression surrogate is fit only on the current training split and evaluated on the validation split. The test split is never accessed. Fitness is

$$F(z) = F1_{val}(z) - \lambda \frac{\|z\|_0}{d} \quad (2)$$

where $\lambda = 0.01$ penalizes unnecessarily large subsets. The population contains 30 chromosomes and evolves for 25 generations using size-3 tournament selection, uniform crossover with probability 0.8, bit-flip mutation at rate $1/d$, two-member elitism, and deterministic repair of empty chromosomes. Surrogate fitting is capped at 20,000 stratified training records for tractability. Every stochastic operation, surrogate, and split receives the experimental seed.

Table 4. Feature Selection Summary for the IDS Datasets

Dataset	Method	Orig. Feat.	Final Feat.	Top Selected Features
UNSW-NB15 [31]	Genetic algorithm	42	11–15	Seed-specific behavioral, state, load, and time-to-live features
CIC-IDS-2017 [32]	Genetic algorithm	78	35–48	Seed-specific packet, timing, flag, window, and flow-rate features

The GA is model-agnostic because its surrogate is neither Random Forest nor XGBoost. Within each seed, all six evaluated models receive the identical GA-selected subset. Selection is repeated inside each seed to preserve the split boundary. RF-SHAP remains only as the ablation in Table 9. The number of selected features varied from 11 to 15 on UNSW-NB15 and from 35 to 48 on CIC-IDS-2017 across the five seeds (Table 4).

3.5 The Self-Supervised Transformer under Study

The proposed SST-IDS model follows a two-stage learning design. The first stage is self-supervised pretraining, where the model learns representations from selected network-flow features using masked feature prediction. The second stage is supervised fine-tuning, where the pretrained model is adapted for binary intrusion detection.

Each traffic record is represented as a vector of selected network-flow features after preprocessing and feature selection. In this study, the Transformer operates over feature tokens rather than temporal packet sequences. That means each selected feature is treated as one token after projection into an embedding space. The model therefore learns relationships among features within a traffic record, not temporal relationships across packet sequences.

Let each input sample be represented as:

$$X = [x_1, x_2, \dots, x_d] \quad (3)$$

where d is the number of selected features. Each feature x_i is projected into an embedding vector:

$$e_i = W_e x_i + b_e \quad (4)$$

where W_e and b_e are learnable embedding parameters. The embedded feature tokens are then passed to the Transformer encoder.

After preprocessing and feature selection, a fixed feature order is maintained across all samples. If feature-index embeddings are used, they represent the fixed index of each selected feature and should not be interpreted as temporal positional encoding. This distinction is important because network-flow features are tabular attributes, not natural language words or time-ordered packet tokens.

3.6 Transformer Encoder

The Transformer encoder models relationships among selected feature tokens. Its self-attention mechanism allows the model to estimate how much each feature token contributes to the representation of other feature tokens.

For query matrix Q , key matrix K , and value matrix V , scaled dot-product attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (5)$$

where d_k is the dimension of the key vectors. Multi-head attention extends this operation by computing attention across several representation subspaces, allowing the model to capture different types of feature relationships.

The output of the attention layer is passed through a feed-forward network with nonlinear activation. Residual connections and layer normalization stabilize training and improve gradient flow. Dropout is used to reduce overfitting. Table 5 reports the main hyperparameter settings used in the SST-IDS model. The fixed configuration in Table 5 is a deliberate, disclosed choice rather than the product of a per-model search: under the fairness protocol of Section 3.9, no model receives tuning attention that the others do not, and the training-budget ablation in Table 8 isolates the sensitivity of the results to the schedule component of this configuration.

3.7 Self-Supervised Pretraining

The self-supervised pretraining stage is based on masked feature prediction. During this stage, a random subset of selected input features is masked, and the model is trained to reconstruct the original values of the masked features using the remaining unmasked features.

Let M be a binary mask vector, where $M_i = 0$ indicates that feature x_i is masked and $M_i = 1$ indicates that the feature is retained. The masked input is defined as:

$$\tilde{X} = X \odot M \quad (6)$$

where $\odot$ denotes element-wise multiplication.

The Transformer encoder processes the masked input and produces hidden representations. A reconstruction head then predicts the original values of the masked features. Since the selected network-flow features are continuous after preprocessing and standardization, the masked feature prediction task is formulated as a regression problem. Therefore, mean squared error is used as the self-supervised pretraining loss:

$$L_{SSL} = \frac{1}{|M_0|} \sum_{i \in M_0} (x_i - \hat{x}_i)^2 \quad (7)$$

where M_0 is the set of masked feature indices, x_i is the true value of the masked feature, and $\hat{x}_i$ is the reconstructed value.

Table 5. SST-IDS hyperparameters used for the reported full-budget results. Early stopping monitors validation loss with patience 5 and restores the best weights.

Hyperparameter	Value Used
Input representation	Feature tokens
Embedding dimension (d_{model})	64
Number of attention heads	4
Number of Transformer encoder blocks	2
Feed-forward hidden dimension	128
Dropout rate	0.1
Masking rate	0.30

Hyperparameter	Value Used
Optimizer	Adam (weight decay 1e-5)
Learning rate	1e-3
Gradient clipping (max norm)	1.0
Batch size	256
Maximum pretraining epochs	20
Maximum fine-tuning epochs	20
Early-stopping patience	5 epochs
Train/validation/test split	72%/8%/20%
Repeated runs (random seeds)	5 on each dataset (0–4)
Samples per run	175,341 (UNSW); 20,000 (CIC)
Classification activation	Sigmoid
Self-supervised loss	Mean squared error
Supervised classification loss	Binary cross-entropy

This pretraining step allows SST-IDS to learn relationships among traffic features without relying only on labeled data.

3.8 Supervised Fine-Tuning

After self-supervised pretraining, SST-IDS is fine-tuned for binary intrusion detection. The reconstruction head is replaced or supplemented with a classification head that outputs the probability that a traffic record is malicious.

Given the Transformer encoder output H , a pooled representation is passed to a binary classification layer:

$$\hat{y} = \sigma(W_c H + b_c) \quad (8)$$

where W_c and b_c are classification parameters, σ is the sigmoid activation function, and $\hat{y}$ is the predicted probability of the attack class.

Binary cross-entropy is used as the supervised classification loss:

$$L_{BCE} = -\frac{1}{n} \sum_{i=1}^n [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \quad (9)$$

where y_i is the true binary label and $\hat{y}_i$ is the predicted probability.

Thus, mean squared error is used for masked feature reconstruction during self-supervised pretraining, while binary cross-entropy is used for supervised benign/malicious classification.

3.9 Baseline Models and Fairness Protocol

SST-IDS is compared with five widely used baselines: Random Forest, XGBoost, a one-dimensional CNN, an LSTM, and TabTransformer. To make the comparison informative rather than promotional, all models are trained and evaluated under identical conditions: the same stratified train/validation/test splits (72%/8%/20%), the same selected feature set per dataset (Section 3.4), and the same preprocessing fitted on the training split only. The Random Forest uses 300 trees with balanced class weights; XGBoost uses 400 trees with max depth 6 and a learning rate 0.1. The supervised neural baselines use the same optimizer, learning rate, 20-epoch supervised budget, and early-stopping rule as SST-IDS fine-tuning; only SST-IDS has the additional 20-epoch masked-feature pretraining stage. We run every configuration with repeated random seeds, and report results as mean $\pm$ standard deviation. No model, including SST-IDS, receives tuning attention that the others do not.

3.10 Cross-Dataset Transfer Protocol

To examine transfer under heterogeneous network conditions, we trained SST-IDS, Random Forest, and XGBoost on CIC-IDS-2017 and tested them, without target-domain fine-tuning, on the official UNSW-NB15 testing split (UNSW_NB15_testing-set.csv, 82,332 records). No UNSW-NB15 data of any kind was used for training, so the comparison is strictly zero-shot; evaluating on the official testing split keeps the transfer evaluation set disjoint from the UNSW-NB15 training subset used in the within-dataset experiments.

Because CIC-IDS-2017 and UNSW-NB15 differ in feature definitions, collection environments, traffic distributions, and attack categories, feature alignment was required before transfer testing. Only behavioral features that could be semantically aligned across the two schemas were retained, specifically flow duration, forward and backward packet counts, forward and backward byte totals, and packet rate; no identity attributes (IP addresses or ports) were used. We then applied the source-domain scaling and fixed feature ordering to the target domain.

This experiment was not to claim complete cross-dataset generalization. It is a controlled, strictly zero-shot comparison of whether any model retains discriminative signal across datasets using only behavioral attributes expected to be more portable across infrastructures than identity-based features.

3.11 Explainability Protocol

Explainability was incorporated after GA feature selection to interpret the trained seed-0 SST-IDS model on its CIC-IDS-2017 test split. Kernel SHAP used 100 seeded background records and explained the first 200 held-out records with 100 samples per explanation. Mean absolute SHAP values provide the global ranking over the GA-selected subset; local waterfall plots show how individual feature values moved representative false-positive and false-negative predictions relative to the explainer's base value. SHAP has no role in the primary selector. RF-SHAP is executed only for the feature-selection ablation in Table 9.

3.12 Deployment-Cost Measurement

Because operational deployment depends on more than detection accuracy, we measure inference latency on the CPU environment described in Section 4.1 for SST-IDS and the two strongest baselines under two identical protocols: strict single-flow calls and one 1000-flow batch reported per flow. We also report the Transformer parameter size. These measurements quantify the cost side of the accuracy–cost trade-off examined in Section 5.

3.13 Evaluation Metrics

The model was evaluated using standard binary classification metrics: accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC).

Accuracy measures the proportion of correct predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (10)$$

Precision measures the proportion of predicted attacks that are truly attacks:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (11)$$

Recall measures the proportion of actual attacks correctly detected:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (12)$$

F1-score provides the harmonic mean of precision and recall:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (13)$$

AUC measures the ability of the model to separate benign and malicious traffic across different decision thresholds. A higher AUC indicates stronger discrimination between the two classes.

Table 6. Within-dataset SST-IDS performance under GA feature selection and the full schedule (mean $\pm$ sample standard deviation over five seeds).

Dataset	Accuracy	Precision	Recall	F1-Score	AUC
UNSW-NB15 [31]	0.9257 $\pm$ 0.0040	0.9572 $\pm$ 0.0207	0.9333 $\pm$ 0.0240	0.9447 $\pm$ 0.0035	0.9781 $\pm$ 0.0076
CIC-IDS-2017 [32]	0.9350 $\pm$ 0.0137	0.7661 $\pm$ 0.0458	0.9592 $\pm$ 0.0297	0.8509 $\pm$ 0.0275	0.9841 $\pm$ 0.0099

3.14 Statistical Testing

Statistical testing was performed to compare SST-IDS with baseline models, including Random Forest, XGBoost, CNN, LSTM, and TabTransformer. The purpose was to determine whether observed performance differences were statistically significant for selected metrics.

A significance threshold of $p \leq 0.05$ was used. Results are reported for both statistically significant and non-significant comparisons to avoid overstating the advantage of SST-IDS. This is important because some model-metric comparisons may show numerical improvement but may not reach statistical significance.

The statistical testing results are reported in Section 4.3.

4 Results

4.1 Experimental Setup

All final experiments ran on a 13th Gen Intel Core i9-13900H CPU (2.60 GHz) with 64 GB RAM, Python 3.12.13, and no GPU. Every model used the seed-specific GA feature set and split described in Section 3. SST-IDS used the full 20+20 schedule in Table 5; the supervised neural baselines used the corresponding 20-epoch supervised budget. In every neural run, early stopping monitored validation loss and restored the best weights. UNSW-NB15 used all 175,341 training records. CIC-IDS-2017 used the disclosed 20,000-flow sample with fixed sampling seed 991. We used five experimental seeds (0–4) on both datasets. Section 5.3 provides the public repository and exact reproduction command.

4.2 Detection Performance under Identical Conditions (RQ1)

Table 6 reports the within-dataset performance of the self-supervised Transformer, and Table 7 places it alongside the five baselines under identical conditions.

Three observations answer RQ1. First, XGBoost has the highest mean accuracy, F1, and AUC on both datasets. Second, SST-IDS remains in the same numerical band as the supervised neural baselines despite its pretraining stage. Third, the near-perfect CIC tree results require caution because of documented benchmark defects and the disclosed 20,000-flow sample. The stronger basis for the headline comparison is therefore the complete UNSW-NB15 experiment.

Table 7. Comparative results under GA feature selection and the full schedule (mean $\pm$ sample standard deviation over five seeds).

Dataset	Model	Accuracy	Precision	Recall	F1	AUC
UNSW-NB15	SST-IDS	0.9257 $\pm$ 0.0040	0.9572 $\pm$ 0.0207	0.9333 $\pm$ 0.0240	0.9447 $\pm$ 0.0035	0.9781 $\pm$ 0.0076
UNSW-NB15	Random Forest	0.9429 $\pm$ 0.0108	0.9585 $\pm$ 0.0131	0.9578 $\pm$ 0.0047	0.9581 $\pm$ 0.0077	0.9821 $\pm$ 0.0133
UNSW-NB15	XGBoost	0.9456 $\pm$ 0.0067	0.9430 $\pm$ 0.0132	0.9796 $\pm$ 0.0058	0.9608 $\pm$ 0.0045	0.9856 $\pm$ 0.0078
UNSW-NB15	CNN	0.9318 $\pm$ 0.0011	0.9446 $\pm$ 0.0191	0.9566 $\pm$ 0.0242	0.9502 $\pm$ 0.0018	0.9750 $\pm$ 0.0079

Dataset	Model	Accuracy	Precision	Recall	F1	AUC
UNSW-NB15	LSTM	0.9269±0.0059	0.9465±0.0202	0.9470±0.0297	0.9463±0.0052	0.9744±0.0098
UNSW-NB15	TabTransformer	0.9296±0.0059	0.9486±0.0235	0.9491±0.0331	0.9482±0.0055	0.9766±0.0076
CIC-IDS-2017	SST-IDS	0.9350±0.0137	0.7661±0.0458	0.9592±0.0297	0.8509±0.0275	0.9841±0.0099
CIC-IDS-2017	Random Forest	0.9955±0.0011	0.9883±0.0046	0.9883±0.0030	0.9883±0.0027	0.9994±0.0005
CIC-IDS-2017	XGBoost	0.9963±0.0007	0.9917±0.0034	0.9894±0.0006	0.9905±0.0017	0.9998±0.0001
CIC-IDS-2017	CNN	0.9322±0.0188	0.7615±0.0602	0.9525±0.0171	0.8451±0.0365	0.9838±0.0050
CIC-IDS-2017	LSTM	0.9284±0.0274	0.8284±0.0809	0.8255±0.2424	0.7979±0.1340	0.9340±0.1151
CIC-IDS-2017	TabTransformer	0.9314±0.0100	0.7608±0.0437	0.9465±0.0465	0.8419±0.0182	0.9801±0.0054

Table 8. SST-IDS training-budget ablation on full UNSW-NB15 (mean ± sample standard deviation over five seeds).

Schedule	Accuracy	Precision	Recall	F1	AUC
Reduced (12+12, batch 4096)	0.9234±0.0054	0.9457±0.0202	0.9425±0.0297	0.9436±0.0050	0.9705±0.0111
Full (20+20, batch 256)	0.9257±0.0040	0.9572±0.0207	0.9333±0.0240	0.9447±0.0035	0.9781±0.0076

Table 9. GA versus RF-SHAP feature-selection ablation using XGBoost (mean ± sample standard deviation over five seeds). RF-SHAP is an ablation only.

Dataset	Selector	Accuracy	Precision	Recall	F1	AUC
UNSW-NB15	GA	0.9456±0.0067	0.9430±0.0132	0.9796±0.0058	0.9608±0.0045	0.9856±0.0078
UNSW-NB15	RF-SHAP	0.9527±0.0035	0.9555±0.0049	0.9760±0.0006	0.9656±0.0025	0.9921±0.0014
CIC-IDS-2017	GA	0.9963±0.0007	0.9917±0.0034	0.9894±0.0006	0.9905±0.0017	0.9998±0.0001
CIC-IDS-2017	RF-SHAP	0.9929±0.0067	0.9905±0.0029	0.9725±0.0343	0.9811±0.0183	0.9994±0.0008

Table 8 closes the training-budget concern on UNSW-NB15. The full schedule raises mean AUC by 0.0076 and yields smaller improvements in accuracy and F1, but it does not overturn the comparison with XGBoost.

Table 9 shows that RF-SHAP gives XGBoost a modest advantage on UNSW-NB15, whereas GA gives it a modest advantage on CIC-IDS-2017. This dataset-dependent result supports the concern that a tree-derived selector can favor tree-aligned structure and justifies the model-agnostic GA as the primary protocol.

4.3 Paired Statistical Validation

Statistical testing sharpens these observations. Table 10 reports paired t-tests over the UNSW-NB15 seeds for every SST-IDS–baseline pair and every metric.

On UNSW-NB15, XGBoost exceeds SST-IDS significantly on accuracy, recall, F1, and AUC, while precision is not distinguishable at this power. Random Forest exceeds SST-IDS significantly on accuracy and F1. Several neural comparisons are also significant, so the earlier blanket claim of neural equivalence is no longer supported under five seeds. On CIC-IDS-2017, neither TabTransformer, CNN, nor LSTM differs significantly from SST-IDS on any reported metric, while both tree ensembles exceed it significantly on accuracy, precision, F1, and AUC. Recall is non-significant for both tree comparisons.

Non-significant findings are interpreted only as not distinguishable at this power, not as evidence of equivalence.

Table 10. Paired t-test p-values for SST-IDS versus each baseline over five seeds. Asterisks indicate $p \leq 0.05$; unmarked values are non-significant.

Dataset	Baseline	Accuracy	Precision	Recall	F1	AUC
UNSW-NB15	TabTransformer	0.0597	0.0148*	0.0302*	0.0504	0.0423*
UNSW-NB15	CNN	0.0284*	0.0110*	0.0101*	0.0234*	0.0211*
UNSW-NB15	LSTM	0.5688	0.0101*	0.0458*	0.3762	0.0636
UNSW-NB15	Random Forest	0.0156*	0.8355	0.0800	0.0183*	0.3367
UNSW-NB15	XGBoost	0.0012*	0.0809	0.0059*	0.0015*	0.0206*
CIC-IDS-2017	TabTransformer	0.7362	0.8792	0.1937	0.6633	0.3370
CIC-IDS-2017	CNN	0.8339	0.9158	0.4175	0.8253	0.9648
CIC-IDS-2017	LSTM	0.6162	0.1415	0.2338	0.4078	0.3513
CIC-IDS-2017	Random Forest	0.0006*	0.0004*	0.1016	0.0004*	0.0267*
CIC-IDS-2017	XGBoost	0.0006*	0.0004*	0.0867	0.0004*	0.0238*

4.4 Zero-Shot Cross-Dataset Transfer (RQ2)

Table 11 reports the strictly zero-shot transfer results: each model is trained on CIC-IDS-2017 and evaluated on UNSW-NB15 without target-domain fine-tuning, using only the semantically aligned behavioral features described in Section 3.

Transfer is weak and unstable. SST-IDS has the highest mean AUC, 0.6126, but its standard deviation is 0.1960, and its thresholded recall and F1 are low. Both tree baselines have mean AUC below 0.5 and effectively predict no positive cases at the fixed threshold. Because the alignment excludes IP addresses and ports, these results cannot be attributed to infrastructure identifiers. RQ2 therefore provides no operational transfer claim. It shows only that SST-IDS retains an inconsistent threshold-free signal that would require target-domain calibration, fine-tuning, or explicit adaptation. Closing this gap is a domain-adaptation problem rather than a model-selection problem. Candidate strategies include fine-tuning the pretrained encoder on a small labeled sample of target-domain flows, re-calibrating output probabilities and decision thresholds on a target validation set, and aligning source and target feature distributions before inference, for example through importance weighting or correlation alignment; none of these adaptations is applied here, by design, so that the comparison remains strictly zero-shot.

Table 11. Zero-shot CIC-IDS-2017 to UNSW-NB15 transfer over aligned behavioral features (mean $\pm$ sample standard deviation over five seeds).

Model	Direction	Accuracy	Precision	Recall	F1	AUC
SST-IDS	CIC→UNSW	0.4509 $\pm$ 0.0261	0.3438 $\pm$ 0.3583	0.0378 $\pm$ 0.0371	0.0676 $\pm$ 0.0666	0.6126 $\pm$ 0.1960
Random Forest	CIC→UNSW	0.4494 $\pm$ 0.0000	0.0000 $\pm$ 0.0000	0.0000 $\pm$ 0.0000	0.0000 $\pm$ 0.0000	0.4443 $\pm$ 0.1631
XGBoost	CIC→UNSW	0.4486 $\pm$ 0.0011	0.3757 $\pm$ 0.2685	0.0004 $\pm$ 0.0004	0.0009 $\pm$ 0.0008	0.3950 $\pm$ 0.0856

4.5 Explainability (RQ3)

Table 12 and Figure 2 report the global SHAP feature-importance magnitudes for the Transformer on CIC-IDS-2017, and Figure 3 shows local waterfall explanations for a representative false positive and false negative.

Table 12. Global SHAP feature-importance magnitudes (mean |SHAP|) over the seed-0 GA-selected CIC-IDS-2017 features

Rank	Feature	Mean SHAP
1	PSH Flag Count	0.0808
2	Bwd Packet Length Std	0.0668
3	Average Packet Size	0.0639
4	Packet Length Mean	0.0539
5	Init_Win_bytes_forward	0.0241
6	Bwd Packet Length Mean	0.0239
7	Destination Port	0.0173
8	min_seg_size_forward	0.0138
9	Bwd IAT Total	0.0125
10	URG Flag Count	0.0112
11	Init_Win_bytes_backward	0.0099
12	Fwd Header Length	0.0057
13	Active Std	0.0057
14	Active Max	0.0049
15	act_data_pkt_fwd	0.0046

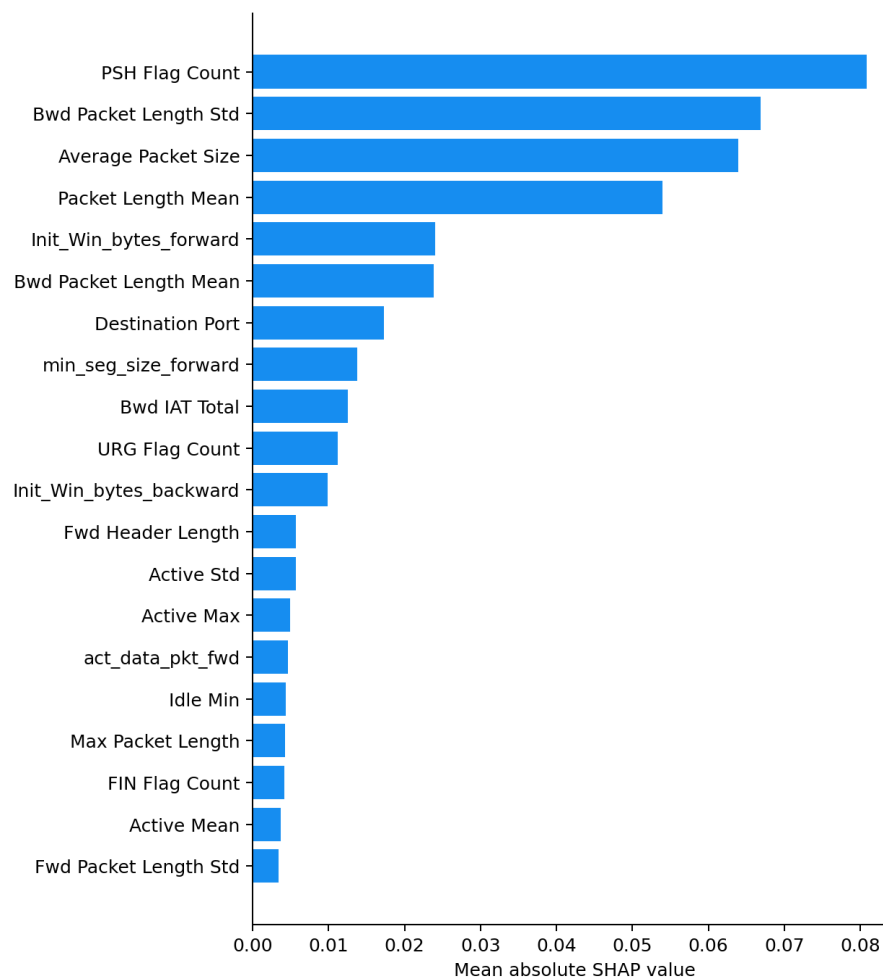


Figure 2. Global SHAP feature importance for SST-IDS on the seed-0 GA subset of CIC-IDS-2017. The largest contributions come from TCP flags, packet-length dispersion, packet size, and window statistics.

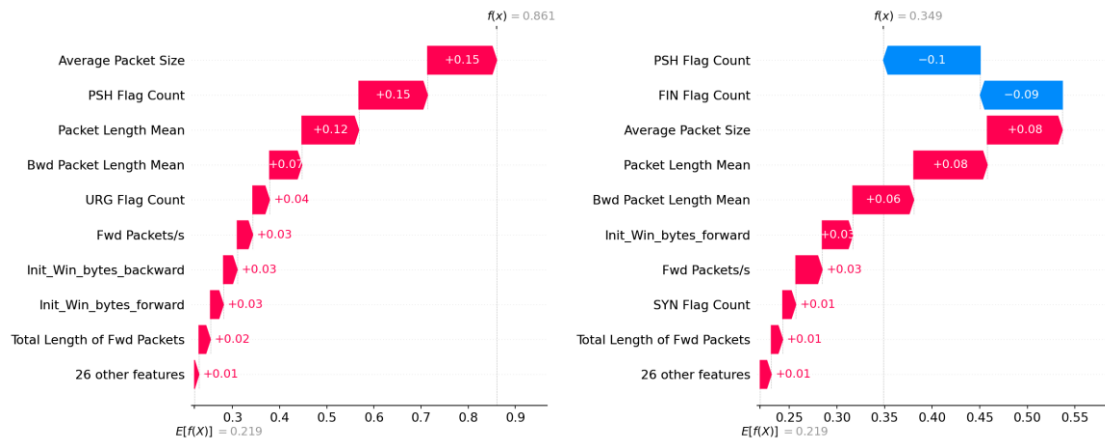


Figure 3. Local SHAP (waterfall) explanations for individual predictions on CIC-IDS-2017: a false positive (left) and a false negative (right). Each plot shows how the selected flow-feature values pushed the prediction toward or away from the malicious class relative to the base value.

The dominant CIC-IDS-2017 attributions are behavioral flow statistics. Destination Port ranks seventh in the displayed seed-0 subset, so the interpretation is not that port information is absent, but that the leading evidence also includes flags, packet-length dispersion, packet size, timing, and window statistics. The flow CSVs contain no source or destination IP-address columns. The local explanations show which values pushed one benign flow over the alert threshold and which values allowed one malicious flow to pass.

4.6 Deployment Costs (RQ3)

Table 13 reports the deployment-cost measurements.

Table 13. Per-flow inference latency on CPU, measured identically for all models under two protocols: amortized over a 1000-flow batch (one predict call per batch) and strict single-flow inference (one predict call per flow), plus model size.

Model	Batched (ms/flow)	Single-flow (ms)	Model size (MB)
SST-IDS	0.0403	0.7724	0.290
Random Forest	0.0758	36.6328	—
XGBoost	0.0015	0.1872	—

The seed-0 Transformer occupies about 0.290 MB. Because latency depends on invocation, Table 13 reports two protocols applied identically to all three models. Amortized over 1000-flow batches, SST-IDS requires 0.0403 ms per flow, between XGBoost at 0.0015 ms and Random Forest at 0.0758 ms. Under strict single-flow calls, the corresponding measurements are 0.7724, 0.1872, and 36.6328 ms. Framework dispatch dominates the strict single-flow measurements. All batched values remain below one millisecond.

5 Discussion

5.1 What is Results Mean

The central empirical message is that full-budget training improves SST-IDS but does not make it the strongest detector on the full UNSW-NB15. XGBoost retains higher mean accuracy, F1, and AUC, with significant paired differences on those metrics. The five-seed tests also show that the neural models are not uniformly equivalent: several UNSW comparisons reach significance, whereas no neural comparison reaches significance on the sampled CIC experiment. The evidence therefore supports model- and dataset-specific statements rather than a blanket equivalence claim.

Three secondary findings qualify the picture. First, zero-shot transfer is unstable and not operational for any model; SST-IDS has the highest mean AUC but also large variance. Second, GA removes tree-

derived feature-selection bias from the primary comparison, while the RF-SHAP ablation confirms that selector choice can change results by dataset. Third, every measured batched inference latency is sub-millisecond, with SST-IDS between XGBoost and Random Forest.

5.2 Recommendations for Practitioners

For binary flow-based detection on tabular features with adequate labels, gradient-boosted trees should be the default: in this study they were the most accurate, the fastest at inference by a wide margin, the cheapest to train (no GPU, no pretraining stage), and directly compatible with SHAP-based explanation. An attention-based detector is currently justified only by requirements the trees cannot meet, such as intended reuse of pretrained representations across tasks or environments, integration of non-tabular inputs, or research settings targeting label-scarce regimes. Teams evaluating vendor or academic claims of Transformer superiority on tabular flow data should ask whether the baselines received equal feature sets, splits, tuning effort, repeated runs, and statistical testing, since in our experience the direction of the conclusion can invert when they do.

5.3 Limitations and Threats to Validity

Several limitations bound the conclusions. Internal validity: the full 20+20 schedule and early stopping address the training-budget concern on complete UNSW-NB15, but the schedule is not an exhaustive hyperparameter search. Construct validity: CIC-IDS-2017 has documented labeling and feature-computation defects [33] that can inflate performance. We remove invalid records, pool captures, use five seeds, and scope the headline claim to UNSW-NB15, but these steps do not repair the benchmark. Computational scope: CIC uses a fixed, stratified 20,000-flow sample due to computational resource constraints on the four-model, five-seed neural suite; the full neural schedule was retained. CPU-only execution is itself a controlled choice, holding hardware identical across all models for training and for the deployment-cost measurements in Section 4.5, but it leaves the GPU-scale behavior of the Transformer uncharacterized. External validity: the study covers two datasets, binary detection, one self-supervised objective, one GA configuration, and no attack-type breakdown, so the findings are scoped to these two benchmarks and should not be extrapolated to other traffic environments or attack distributions without re-evaluation. Statistical interpretation: five seeds improve power but do not establish equivalence; non-significant findings mean only not distinguishable at this power. Transfer variance is high and rules out an operational generalization claim. These boundaries also define the future-work agenda: evaluating the protocol on additional recent IDS benchmarks, extending the binary formulation to multi-class attack categorization, and adding a GPU-based replication to characterize Transformer scalability on larger samples.

6 Conclusion

This study integrated model-agnostic GA feature selection into a controlled comparison of SST-IDS and five baselines. Five-seed, full-budget results on complete UNSW-NB15 show that adequate training improves SST-IDS but does not reverse XGBoost's advantage in accuracy, F1, and AUC. The sampled CIC experiment provides corroborating evidence but is explicitly down-weighted because of dataset defects and sampling. Zero-shot transfer is unstable for all three tested models and supports no deployment claim. SHAP and deployment measurements provide traceable explanations and like-for-like costs. Every reported value is generated from seed-level JSON results by the supplied reproduction workflow. Future work will extend the protocol to additional recent IDS benchmarks and multi-class attack categorization, with a GPU-based replication on larger samples.

DECLARATIONS

Funding. The authors received no external funding for this research.

Conflicts of interest. The authors declare no competing interests.

Ethics approval. Ethics approval was not required for this study, which used only publicly available benchmark network-traffic datasets and did not involve human participants or animals.

Informed consent. Not applicable; the study did not involve human participants.

Data availability. The datasets analysed in this study are publicly available from their official providers (UNSW-NB15 and CIC-IDS-2017).

Author contributions. Mohammad F. Al-Hammouri: Conceptualization, Methodology, Supervision, Writing – review and editing. Bandi Vamsi: Methodology, Investigation, Validation, Writing – original draft, Writing – review and editing. Muder M. AlMi'ani: Methodology, Formal analysis, Visualization, Writing – review and editing. Elvira Al Bataineh: Methodology, Data curation, Validation, Writing – review and editing. Ali Al Bataineh: Conceptualization, Methodology, Software, Investigation, Formal analysis, Writing – original draft, Writing – review and editing, Supervision. All authors read and approved the final version of the manuscript.

Use of artificial intelligence. OpenAI ChatGPT was used to assist with language editing and improve the clarity and readability of portions of this manuscript. The authors reviewed and verified all content and take full responsibility for the technical methods, analyses, results, figures, code, references, and conclusions. No generative AI output was used as a source of scientific ideas, experimental results, or references.

Acknowledgements. This work was supported by the Gulf University for Science and Technology, Kuwait.

REFERENCES

- [1] M. M. Rahman, S. Al Shakil, and M. R. Mustakim, "A Survey on Intrusion Detection System in IoT Networks," *Cyber Security and Applications*, vol. 3, p. 100082, 2025, doi: 10.1016/j.csa.2024.100082.
- [2] L. Diana, P. Dini, and D. Paolini, "Overview on Intrusion Detection Systems for Computers Networking Security," *Computers*, vol. 14, no. 3, p. 87, 2025, doi: 10.3390/computers14030087.
- [3] N. Dash, S. Chakravarty, A. K. Rath, N. C. Giri, K. M. AboRas, and N. Gowtham, "An optimized LSTM-based deep learning model for anomaly network intrusion detection," *Scientific Reports*, vol. 15, no. 1, p. 1554, 2025, doi: 10.1038/s41598-025-85248-z.
- [4] V. Radhakrishnan, N. Kabilan, V. Ravi, and V. Sowmya, "Unsupervised Representation Learning Approach for Intrusion Detection in the Industrial Internet of Things Network Environment," in *Analytics Modeling in Reliability and Machine Learning and Its Applications*. Cham, Switzerland: Springer Nature, 2025, pp. 41–76, doi: 10.1007/978-3-031-72636-1_3.
- [5] G. Apruzzese, M. Andreolini, L. Ferretti, M. Marchetti, and M. Colajanni, "Modeling realistic adversarial attacks against network intrusion detection systems," *Digital Threats: Research and Practice*, vol. 3, no. 3, pp. 1–19, 2022, doi: 10.1145/3469659.
- [6] A. Thaljaoui, "Intelligent network intrusion detection system using optimized deep CNN-LSTM with UNSW-NB15," *International Journal of Information Technology*, pp. 1–17, 2025, doi: 10.1007/s41870-025-02416-0.
- [7] A. A. Megantara and T. Ahmad, "A hybrid machine learning method for increasing the performance of network intrusion detection systems," *Journal of Big Data*, vol. 8, no. 1, p. 142, 2021, doi: 10.1186/s40537-021-00531-w.
- [8] H. Najafi Mohsenabad and M. A. Tut, "Optimizing cybersecurity attack detection in computer networks: a comparative analysis of bio-inspired optimization algorithms using the CSE-CIC-IDS 2018 dataset," *Applied Sciences*, vol. 14, no. 3, p. 1044, 2024, doi: 10.3390/app14031044.
- [9] Z. Abou El Houda, B. Brik, and S. M. Senouci, "A novel IoT-based explainable deep learning framework for intrusion detection systems," *IEEE Internet of Things Magazine*, vol. 5, no. 2, pp. 20–23, 2022, doi: 10.1109/IOTM.005.2200028.
- [10] M. Udurume, V. Shakhov, and I. Koo, "Comparative analysis of deep convolutional neural network-bidirectional long short-term memory and machine learning methods in intrusion detection systems," *Applied Sciences*, vol. 14, no. 16, p. 6967, 2024, doi: 10.3390/app14166967.
- [11] H. Zouhri, A. Idri, and A. Ratnani, "Evaluating the impact of filter-based feature selection in intrusion detection systems," *International Journal of Information Security*, vol. 23, no. 2, pp. 759–785, 2024, doi: 10.1007/s10207-023-00767-y.
- [12] K. C. Santos, R. S. Miani, and F. de Oliveira Silva, "Evaluating the impact of data preprocessing techniques on the performance of intrusion detection systems," *Journal of Network and Systems Management*, vol. 32, no. 2, p. 36, 2024, doi: 10.1007/s10922-024-09813-z.
- [13] S. Kumar, S. Gupta, and S. Arora, "Research trends in network-based intrusion detection systems: A review," *IEEE Access*, vol. 9, pp. 157761–157779, 2021, doi: 10.1109/ACCESS.2021.3129775.
- [14] T. S. Naseri and F. S. Gharehchopogh, "A feature selection based on the farmland fertility algorithm for improved intrusion detection systems," *Journal of Network and Systems Management*, vol. 30, no. 3, p. 40, 2022, doi: 10.1007/s10922-022-09653-9.
- [15] F. Zeng, M. Chen, C. Qian, Y. Wang, Y. Zhou, and W. Tang, "Multivariate time series anomaly detection with adversarial transformer architecture in the Internet of Things," *Future Generation Computer Systems*, vol. 144, pp. 244–255, 2023, doi: 10.1016/j.future.2023.02.015.
- [16] M. Alotaibi, H. A. Mengash, H. Alqahtani, A. M. Al-Sharafi, A. E. Yahya, S. R. Alotaibi, A. O. Khadidos, and A. Yafaz, "Hybrid GWQBBA Model for Optimized Classification of Attacks in Intrusion Detection System," *Alexandria Engineering Journal*, vol. 116, pp. 9–19, 2025, doi: 10.1016/j.aej.2024.12.057.
- [17] A. Reed, L. Dooley, and S. Kouadri, "SA-IDS: A single attribute intrusion detection system for Slow DoS attacks in IoT networks," *Internet of Things*, p. 101512, 2025, doi: 10.1016/j.iot.2025.101512.

- [18] K. Barik and S. Misra, "IDS-Anta: An open-source code with a defense mechanism to detect adversarial attacks for intrusion detection system," *Software Impacts*, vol. 21, p. 100664, 2024, doi: 10.1016/j.simpa.2024.100664.
- [19] D. Natsos and A. L. Symeonidis, "Transformer-based malware detection using process resource utilization metrics," *Results in Engineering*, p. 104250, 2025, doi: 10.1016/j.rineng.2025.104250.
- [20] M. A. Umar, Z. Chen, K. Shuaib, and Y. Liu, "Effects of feature selection and normalization on network intrusion detection," *Data Science and Management*, vol. 8, no. 1, pp. 23–39, 2025, doi: 10.1016/j.dsm.2024.08.001.
- [21] N. AL-Nomasy, A. Alamri, A. Aljuhani, and P. Kumar, "Transformer-Based Knowledge Distillation for Explainable Intrusion Detection System," *Computers & Security*, vol. 154, p. 104417, 2025, doi: 10.1016/j.cose.2025.104417.
- [22] S. Ullah, J. Ahmad, M. A. Khan, M. S. Alshehri, W. Boulila, A. Koubaa, et al., "TNN-IDS: Transformer neural network-based intrusion detection system for MQTT-enabled IoT Networks," *Computer Networks*, vol. 237, p. 110072, 2023, doi: 10.1016/j.comnet.2023.110072.
- [23] P. Sarantos, J. Violos, and A. Leivadreas, "Enabling semi-supervised learning in intrusion detection systems," *Journal of Parallel and Distributed Computing*, vol. 196, p. 105010, 2025, doi: 10.1016/j.jpdc.2024.105010.
- [24] C. S. Kalutharage, X. Liu, and C. Chrysoulas, "Neurosymbolic learning and domain knowledge-driven explainable AI for enhanced IoT network attack detection and response," *Computers & Security*, vol. 151, p. 104318, 2025, doi: 10.1016/j.cose.2025.104318.
- [25] A. K. Rai, D. K. Verma, and R. K. Dwivedi, "RTAD-HIS: Regulated Transformer Architecture Based Anomaly Detection Framework Towards Security in Healthcare IoT Systems," *Applied Soft Computing*, vol. 177, p. 113209, 2025, doi: 10.1016/j.asoc.2025.113209.
- [26] Y. Li, G. Chen, and Z. Dong, "Multi-view graph contrastive representative learning for intrusion detection in EV charging station," *Applied Energy*, vol. 385, p. 125439, 2025, doi: 10.1016/j.apenergy.2025.125439.
- [27] S. Hossain, S. M. Senouci, B. Brik, and A. Boualouache, "A privacy-preserving Self-Supervised Learning-based intrusion detection system for 5G-V2X networks," *Ad Hoc Networks*, vol. 166, p. 103674, 2025, doi: 10.1016/j.adhoc.2024.103674.
- [28] X. Gu, G. Howells, and H. Yuan, "A semi-supervised soft prototype-based autonomous fuzzy ensemble system for network intrusion detection," *Applied Soft Computing*, vol. 170, p. 112719, 2025, doi: 10.1016/j.asoc.2025.112719.
- [29] P. Prasad, M. Tahir, and J. Isoaho, "Enhancing Explainability of Artificial Intelligence for Threat Detection in SDN-based Multicast Systems," *Procedia Computer Science*, vol. 257, pp. 569–574, 2025, doi: 10.1016/j.procs.2025.03.073.
- [30] F. Ullah, S. Ullah, G. Srivastava, and J. C. W. Lin, "IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic," *Digital Communications and Networks*, vol. 10, no. 1, pp. 190–204, 2024, doi: 10.1016/j.dcan.2023.03.008.
- [31] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in *Proc. 2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6, doi: 10.1109/MilCIS.2015.7348942.
- [32] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization," in *Proc. 4th Int. Conf. Information Systems Security and Privacy (ICISSP)*, Jan. 2018, pp. 108–116, doi: 10.5220/0006639801080116.
- [33] G. Engelen, V. Rimmer, and W. Joosen, "Troubleshooting an Intrusion Detection Dataset: The CICIDS2017 Case Study," in *Proc. 2021 IEEE Security and Privacy Workshops (SPW)*, 2021, pp. 7–12, doi: 10.1109/SPW53761.2021.00009.
- [34] R. Shwartz-Ziv and A. Armon, "Tabular data: Deep learning is not all you need," *Information Fusion*, vol. 81, pp. 84–90, 2022, doi: 10.1016/j.inffus.2021.11.011.
- [35] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?" in *Advances in Neural Information Processing Systems 35 (NeurIPS 2022), Datasets and Benchmarks Track*, 2022, pp. 507–520, doi: 10.52202/068431-0037.