

A Domain-Aware, Explainable Hybrid MLP-LSTM Framework for Scalable Network Intrusion Detection with Cross-Domain Generalization

Dr. Mohammad Islam

Associate Professor, Department of Computer Science,
Maulana Azad University, Jodhpur-342802 Rajasthan, India

ORCID ID: <https://orcid.org/0009-0001-2548-4262> Email: islamjodhpur@gmail.com

Abstract

Network Intrusion Detection Systems (NIDS) must be able to process high volume, high dimensional traffic, and be understandable by security analysts, and be capable of adapting to new networks. Current hybrid deep learning IDSs generally show high accuracy on a single benchmark dataset, and do not always provide a theoretical basis for feature selection based on the knowledge of network-security, seldom perform a generalization analysis on datasets collected under different conditions, and tend to view explainability as an after-thought tool rather than as an analysis tool. In this paper, a domain-aware, explainable hybrid Multi-Layer Perceptron-Long Short-Term Memory (MLP-LSTM) approach for scalable network intrusion detection is proposed. To handle large amounts of traffic, raw flow-level traffic records are fed into and cleaned with a distributed data processing pipeline (Apache Spark). Features are clustered into four sets based on their motivation for network security: flow-based, protocol-based, time-based and payload-based, and filtered for duplication within each set of features, not across all features in an undifferentiated correlation cut. This produces a feature set which is fed to a hybrid architecture comprising of an MLP branch to model static flow statistics and an LSTM branch to model the temporal traffic behaviour, with the two branches combined using a learned fusion layer rather than simple concatenation and provided with an attention-branch mechanism. SHapley Additive exPlanations (SHAP) are used to explain model decisions, and feature importance is aggregated by group and by attack-type, both of which are domain-aware, to generate security-analyst-relevant explanations. The group-level SHAP analysis points in the direction of further analysis, as it is found that the output of the trained model is practically entirely driven by the flow-based features, with 33 of a total 35 non-flow based features having exactly zero attribution despite high variance in the features themselves. The framework is tested in-domain and, to test generalization, without retraining in a cross-domain scenario, where the model is trained on a first intrusion detection dataset and tested in a second, separately collected dataset. For the proposed model, accuracy of 99.81% in domain and macro-F1 score of 99.82% are achieved over seven traffic categories with CICIDS2017 (Domain A). However, when evaluated without retraining on CSE-CIC-IDS2018 (Domain B), the accuracy drops to 90.46% but macro-F1 drops significantly to 32.22% with a per-class breakdown of this drop almost entirely from a complete failure to detect DDoS traffic cross-domain (0.00% recall on 36,641 true DDoS samples, all classified as DoS or Normal Traffic) while Normal Traffic is still detected well (97.61% recall). The proposed fusion mechanism is evaluated using ablation study, with attention-based fusion achieving 99.82% macro-F1, which is higher than 99.48% macro-F1 of the otherwise identical plain-concatenation hybrid, 99.44% macro-F1 of LSTM-only branch and 96.69% macro-F1 of MLP-only branch. Experiments with data volume scaling demonstrate that inference time grows sub-linearly with the number of training samples (from 10% to 100% of the data, it changes from 6.8s to 37.8s), but remains near-constant per sample (from 0.18-0.20 ms per sample), suggesting it is suitable for near-real-time deployment. The results show that the proposed attention-fusion hybrid model can be very effective for in-domain accuracy, but it is difficult to achieve an effective cross-domain generalization performance when using hybrid MLP-LSTM models for intrusion detection, which highlights the needs for testing cross-domain generalization performance in intrusion detection beyond an in-domain setting. The proposed approach introduces a network-security-based feature engineering method, an attention-based fusion method for hybrid spatial-temporal intrusion detection, a group-and-attack-type explainability analysis, and the experimental results of cross-domain generalization for hybrid MLP-LSTM intrusion detection systems.

Keywords: Network Intrusion Detection; Hybrid Deep Learning; MLP-LSTM; Explainable AI; SHAP; Big Data; Apache Spark; Cross-Domain Generalization; Attention Mechanism

1. Introduction

This is because network traffic today is so large and fast that relying solely on a signature-based approach to detecting intrusions is no longer sufficient to detect novel and evolving patterns. Machine Learning (ML) and recently Deep Learning (DL) have been extensively studied for Network Intrusion Detection Systems (NIDS) that are able to detect complex, non-linear relationships between the features of the traffic and the behaviour of the attacks without the need for manually created signatures. The multi-layer perceptrons (MLP) have demonstrated a potential for static feature representation while the long short term memory (LSTM) networks have proven to be effective in modelling the temporal patterns of network traffic, which makes hybrid architecture a promising solution.

However, there are three practical problems that restrict the use of the current hybrid MLP-LSTM intrusion detection models. First, feature selection is often done based solely on statistical criteria (such as a global correlation threshold or generic feature-importance ranking), without taking into account the domain knowledge of the network-security field, the kinds of features that are diagnostic for the kinds of attack one is interested in. Second, in the cases where explainability is added, it is typically only expressed as a single feature-importance graph that gives no idea of which types of evidence contributed to a given type of alert. Third and most importantly, the vast majority of published models are trained and tested on a single dataset, and reported accuracy may well apply only to that specific dataset, and not a true generalisation of the detection ability; other models that are evaluated on traffic captured under different circumstances have been known to perform significantly worse.

To fill these gaps, this paper proposes a domain-aware, explainable, and scalable hybrid MLP-LSTM model. The novel contributions of this work are:

- Domain-aware feature engineering strategy grouping the flow-level features into three categories, namely flow, protocol, time and payload according to the network-security semantics, where redundancy filtering is achieved within each category.
- A fusion mechanism for combining MLP (spatial/static) and LSTM (temporal) branch representations that is based on attention, rather than concatenation as in previous hybrid architectures.
- A distributed preprocessing pipeline on Apache Spark, to ingest and clean high dimensional, large-scale traffic data, along with an empirical scalability study as the data volume grows.
- An explainability analysis based on SHAP, which can group features by domain knowledge and calculate feature importance for each attack type, instead of a single undifferentiated importance ranking, resulting in explanations that are relevant to the analyst.
- A cross-domain evaluation protocol, which involves evaluating the trained model on a second, separately-collected intrusion detection dataset, without retraining, and directly demonstrating the generalization capabilities across the training distribution.

The rest of this paper is organized as follows. Section 2 looks at the related work in hybrid deep learning intrusion detection systems, explainability in NIDS and cross-dataset evaluation. The proposed framework is detailed described in Section 3. Section 4 provides experimental setup including datasets, implementation and evaluation metrics. The results are reported and discussed in Section 5. Some limitations and threats to validity are presented in Section 6. The paper is ended in Section 7 and directions for future work are outlined.

2. Related Work

2.1 Hybrid Deep Learning Models for Intrusion Detection

A substantial body of work has explored combining feedforward and recurrent architectures for intrusion detection, typically pairing convolutional or dense layers for spatial feature extraction with LSTM or GRU layers for temporal modelling of flow sequences, a design pattern surveyed broadly by Mohammadpour et al. across CNN-based network intrusion detection approaches [1]. Representative recent examples include Halbouni et al.'s CNN-LSTM model, which reports 99.60% accuracy on CICIDS2017 [2]; Phalaagae et al.'s attention-augmented CNN-LSTM for IoT

sensor networks, which reports 98.88–99.03% accuracy and 99.36% F1 on the same dataset [3]; and Hossan et al.'s CNN-LSTM evaluation on CSE-CIC-IDS2018, which reports 99.10% accuracy [4]. These hybrid approaches consistently outperform single-branch baselines on benchmark datasets such as NSL-KDD [5], UNSW-NB15 [6], CICIDS2017, and CICIDS2018, motivating the continued study of hybrid designs. However, in each of these studies branch outputs are merged through simple concatenation (or, in [3], attention is applied to weight individual input features ahead of a separate feature-selection step, rather than to fuse two branches), and none report evaluation beyond a single-dataset, in-domain train/test split — gaps this paper's design directly targets (Section 2.5).

2.2 Explainability in Network Intrusion Detection

Explainable AI (XAI) methods, most notably SHAP and LIME, have increasingly been applied to intrusion detection models to address the black-box nature of deep learning classifiers and to support analyst trust. Mane and Rao were among the first to apply SHAP, LIME, and related model-agnostic explainers to a deep neural network IDS on NSL-KDD, establishing feature-attribution scores as a standard reporting format for intrusion detection explainability [7]. Existing work in this vein typically reports a single global feature-importance summary. Comparatively little attention has been paid to structuring explanations around domain-relevant feature categories or to examining how feature importance shifts across different attack types, both of which are more directly actionable for a security operations context.

2.3 Big Data Processing for Intrusion Detection

Distributed processing frameworks such as Apache Spark and Hadoop have been used to scale intrusion detection pipelines to high-volume traffic, primarily for preprocessing and feature extraction stages ahead of model training. Hagar and Gawali, for instance, combine Apache Spark with CNN and LSTM models for high-performance intrusion detection on CSE-CIC-IDS2018 — the same dataset used as Domain B in this study — reporting feature-count reduction from 84 to 19 via random-forest selection and strong per-class accuracy on the resulting Spark-based pipeline [8]. This work follows a similar preprocessing pattern, using Spark for large-scale ingestion and cleaning, but additionally reports an empirical, data-volume-based scalability analysis of the downstream model, which prior Spark-based IDS studies typically do not quantify.

2.4 Cross-Dataset Generalization

Most intrusion detection studies train and evaluate on a single dataset with an in-distribution train/test split. A smaller number of studies have examined cross-dataset performance and consistently report substantial accuracy degradation when a model trained on one dataset is evaluated on another. Cantone et al., in the study most directly comparable to the cross-domain protocol used here, train four machine learning classifiers on CIC-IDS-2017 and CSE-CIC-IDS2018 (the same dataset pair used as Domain A and Domain B in this paper) and report near-perfect within-dataset accuracy collapsing to performance "largely commensurate with random chance" under cross-dataset evaluation for most classifier-dataset combinations [9]. This underscores that in-domain accuracy alone overstates real-world detection capability. This work adopts a cross-domain evaluation protocol as a core, rather than supplementary, component of the proposed framework.

2.5 Summary of Gaps Addressed

Table 1: Summary of gaps in typical prior work addressed by this study.

Aspect	Typical prior work	This work
Feature selection	Global statistical filtering / generic importance ranking	Domain-aware grouping (flow/protocol/time/payload) with within-group redundancy filtering
Branch fusion	Simple concatenation of MLP and LSTM outputs	Learned attention-based fusion

Aspect	Typical prior work	This work
Explainability	Single global SHAP/LIME summary plot	SHAP aggregated by domain-aware group and by attack type
Evaluation	Single-dataset train/test split	In-domain evaluation plus cross-domain evaluation on an independent dataset
Scalability	Rarely quantified	Data-volume-based training time and inference latency benchmark

3. Proposed Methodology

The proposed framework consists of five stages, illustrated conceptually in Figure 1: (i) distributed data ingestion and cleaning using Apache Spark; (ii) domain-aware feature engineering; (iii) an attention-fusion hybrid MLP-LSTM classifier; (iv) SHAP-based explainability analysis; and (v) cross-domain evaluation. Each stage is described in the following subsections.



Figure 1: End-to-end framework architecture — Spark Preprocessing → Domain-Aware Feature Engineering → Attention-Fusion MLP-LSTM → SHAP Explainability → Cross-Domain Evaluation.

3.1 Distributed Data Preprocessing (Apache Spark)

Raw flow-level traffic records are stored as multiple large CSV files, a common distribution format for benchmark intrusion detection datasets generated with CICFlowMeter. These files are ingested using Apache Spark [10] in local cluster mode, which parallelizes reading and cleaning across the available compute cores. Preprocessing performs the following operations: (1) normalization of column names to resolve whitespace and formatting inconsistencies across files; (2) casting of numeric feature columns to a consistent floating-point type; (3) replacement of infinite values, which occur in ratio-based features such as flow bytes-per-second when flow duration is zero, with null and subsequent removal of affected rows; and (4) removal of exact duplicate records. The use of Spark at this stage supports scalability to traffic volumes that exceed the practical limits of single-threaded, in-memory pandas processing, and processing time at this stage is reported as part of the scalability analysis in Section 5.

3.2 Domain-Aware Feature Engineering

Rather than using the full set of approximately 78 CICFlowMeter-derived features directly, or applying a single undifferentiated statistical feature-selection procedure, features are first organized into four categories motivated by network-security semantics:

- Flow-based features: flow duration, forward/backward packet and byte counts, flow-level byte and packet rates, and flow inter-arrival-time statistics, which characterize the overall shape of a connection.

- Protocol-based features: transport-layer protocol identifier and TCP flag counts (SYN, ACK, FIN, RST, PSH, URG, ECE, CWE), which are directly diagnostic for protocol-abuse attacks such as SYN floods and port scans.
- Time-based features: forward/backward inter-arrival-time statistics and active/idle period statistics, which capture temporal behaviour such as beaconing or burstiness associated with botnet and command-and-control traffic.
- Payload-based features: forward/backward packet length statistics and average packet size, which are informative for payload-heavy attacks such as data exfiltration and certain web-based attacks.

In each group, a correlation-based redundancy filter is applied (drop a feature when the absolute Pearson correlation is larger than 0.95 with another feature in the same group that has already been kept). The filtering, instead of across the entire feature set, guarantees that when features in different categories are correlated with each other, there is at least one representative of each category left in the resulting set. Any features that are not assigned to any of the four categories will be included in a catch-all, residual feature set to avoid any information being lost silently. The domain-aware feature set generated is passed on to the two branches of the hybrid classifier in Section 3.3 and directly fed into the explainability analysis in Section 3.4.

3.3 Attention-Fusion Hybrid MLP-LSTM Architecture

The classifier consists of two parallel branches followed by an attention-based fusion layer and a classification head. The MLP branch receives the domain-aware feature vector for a single flow record and applies two dense layers (128 and 64 units, ReLU activation) with batch normalization and dropout, producing a 64-dimensional static representation. The LSTM branch [11] receives a sliding window of T consecutive flow records over the same feature set and applies a single LSTM layer (128 units) followed by a dense layer (64 units, ReLU activation) with dropout, producing a 64-dimensional temporal representation.

Rather than concatenating the two 64-dimensional branch representations directly, the two vectors are stacked and passed through a small attention sub-network [12]: a shared dense layer with tanh activation produces a scalar compatibility score per branch, a softmax normalizes the two scores into attention weights that sum to one, and the branch representations are combined as a weighted sum using these learned weights. This allows the model to up-weight the static (MLP) or temporal (LSTM) representation on a per-instance basis, rather than treating both branches as equally informative for every traffic pattern. The fused representation is passed through a final dense layer and a softmax output layer over the attack classes present in the training data.

3.4 Explainability via SHAP

Model decisions are explained using SHapley Additive exPlanations (SHAP) [13], which attributes each prediction to individual input features based on cooperative game-theoretic Shapley values. Because SHAP's model-agnostic kernel explainer is computationally expensive on deep models, explanations are computed on a representative background sample and a held-out evaluation sample rather than the full test set. In addition to the standard global feature-importance summary, mean absolute SHAP values are aggregated in two further ways: (i) by domain-aware feature group (flow-based, protocol-based, time-based, payload-based, other), producing a group-level importance profile, and (ii) by attack type, producing a group-level importance profile conditioned on each class predicted by the model. This yields explanations of the form "for Denial-of-Service traffic, protocol-based features contribute the majority of the evidence," which are more directly interpretable by a security analyst than an undifferentiated ranked list of individual features.

3.5 Cross-Domain Evaluation Protocol

To assess whether the framework generalizes beyond the statistical characteristics of a single collection environment, the model is evaluated under two protocols. In the in-domain protocol, the model is trained and evaluated on disjoint train/validation/test splits drawn from a single dataset (Domain A). In the cross-domain protocol, the model trained only on Domain A, with no retraining or fine-tuning is evaluated on a second, independently collected dataset (Domain B). Domain B is passed through the identical Spark cleaning pipeline and domain-aware feature selection,

and is scaled using the feature scaler fit exclusively on Domain A, so that no information from Domain B leaks into preprocessing. Only records whose label also occurs in Domain A's label set are retained for cross-domain evaluation, since the classification head cannot predict classes unseen during training. The gap between in-domain and cross-domain performance is reported as a direct, quantitative measure of generalization.

4. Experimental Setup

4.1 Datasets

Experiments use two independently collected, publicly available intrusion detection benchmark datasets, both generated using CICFlowMeter and distributed by the Canadian Institute for Cybersecurity (CIC): CICIDS2017 as Domain A (used for training, validation, and in-domain testing) and CSE-CIC-IDS2018 as Domain B (used exclusively for cross-domain evaluation, with no exposure during training) [14]. CSE-CIC-IDS2018 was produced under the same collaborative CIC project as CICIDS2017 jointly with the Communications Security Establishment, using a later, larger AWS-based network topology and CIC's own citation guidance for both releases points to the same foundational publication [14]. After Spark-based cleaning, domain-aware feature selection, SMOTE-based class-imbalance mitigation, and sequence windowing (Section 3), the resulting Domain A pool was split 72/8/20 into training, validation, and test partitions: the training partition used for the full (100%-data) model comprised 657,826 samples, and the held-out test partition on which all in-domain metrics in Section 5.1 are reported comprised 182,730 samples across the seven attack categories listed below. For cross-domain evaluation, Domain B was drawn from the full 7.89-million-row CSE-CIC-IDS2018 collection and restricted to samples whose label had a confident mapping to a Domain A category (Section 3.5), yielding 500,062 evaluated samples across two categories (Table below).

Table 2: Domain A (CICIDS2017) test-partition class distribution.

Domain A (CICIDS2017) — test partition	Samples
Bots	16,611
Brute Force	16,612
DDoS	16,612
DoS	16,612
Normal Traffic	83,059
Port Scanning	16,612
Web Attacks	16,612
Total (test partition)	182,730

Table 3: Domain B (CSE-CIC-IDS2018) cross-domain evaluation subset class distribution.

Domain B (CSE-CIC-IDS2018) — cross-domain evaluation subset	Samples
DDoS	36,641
Normal Traffic	463,421
Total (confidently mapped subset)	500,062

The domain-aware feature selection described in Section 3.2 yielded a final feature set of 44 features after within-group correlation filtering (threshold 0.95), broken down by group in the table below.

Table 4: Domain-aware feature counts before and after within-group correlation filtering.

Feature group	Before filtering	After filtering (final)
Flow-based	9	9
Protocol-based	3	3
Time-based	14	12
Payload-based	11	7
Other (unmapped)	15	13
Total	52	44

[Note: the validation partition (~8% of the post-SMOTE Domain A pool, used only for early stopping) is omitted above as its exact size was not separately logged.]

4.2 Implementation Details

- Preprocessing: Apache Spark (local mode) for ingestion/cleaning; StandardScaler (fit on Domain A only) for feature scaling; SMOTE [15] for class-imbalance mitigation, with per-class oversampling capped relative to the majority class to control dataset growth.
- Sequence construction: sliding windows of $T = 5$ consecutive records for the LSTM branch, paired with the corresponding final record's flat feature vector for the MLP branch.
- Architecture: MLP branch (Dense 128 → Dense 64, ReLU, BatchNorm, Dropout 0.3); LSTM branch (LSTM 128 → Dense 64, ReLU, Dropout 0.3); attention fusion (Dense/tanh compatibility scoring → softmax weighting → weighted sum); classification head (Dense 64 → Softmax).
- Training: Adam optimizer, categorical cross-entropy loss, batch size 512, up to 30 epochs with early stopping (patience 5) and learning-rate reduction on plateau, 70/10/20 train/validation/test split (stratified) on Domain A.
- Explainability: SHAP KernelExplainer with a 100-instance background sample and a 200-instance evaluation sample.

- Environment: Kaggle Notebooks (Python, TensorFlow/Keras, PySpark, SHAP, imbalanced-learn), [GPU accelerator model to be confirmed].

4.3 Evaluation Metrics

Classification performance is reported using accuracy, macro-averaged precision, macro-averaged recall, and macro-averaged F1-score; macro averaging is used, in addition to overall accuracy, because intrusion detection datasets are highly class-imbalanced and macro averaging weights minority attack classes equally with the majority class. Per-class precision/recall/F1 and confusion matrices are reported for detailed error analysis. Scalability is quantified using training time and per-sample inference latency as a function of training data volume (10%, 25%, 50%, and 100% of the available training data). Cross-domain generalization is quantified as the difference between in-domain and cross-domain accuracy and macro-F1.

5. Results and Discussion

This section reports experimental results. All tables below are structured to match the outputs produced by the accompanying notebook; populate them with the printed metrics after execution.

5.1 In-Domain Classification Performance

Table 5: In-domain classification performance — proposed model vs. ablation baselines.

Model	Accuracy	Macro-Precision	Macro-Recall	Macro-F1
MLP only (baseline)	96.63%	95.89%	97.62%	96.69%
LSTM only (baseline)	99.42%	99.16%	99.74%	99.44%
MLP-LSTM (concatenation, baseline)	99.46%	99.42%	99.53%	99.48%
Proposed: Domain-Aware Attention-Fusion MLP-LSTM	99.81%	99.79%	99.85%	99.82%

The proposed model reaches 99.81% accuracy, 99.79% macro-precision, 99.85% macro-recall, and 99.82% macro-F1 on the in-domain CICIDS2017 test split (182,730 samples) across seven traffic categories (Normal Traffic, Bots, Brute Force, DDoS, DoS, Port Scanning, and Web Attacks), and is the strongest of all four architectures compared. Training converged smoothly: validation loss decreased from 0.014 after the first epoch to 0.0036 by the final epoch, and training and validation accuracy remained closely matched throughout training, indicating no substantial overfitting under the learning-rate schedule and early-stopping configuration used.

Per-class results (Table below) show the model performs near-perfectly on five of the seven categories (Bots, Brute Force, DDoS, DoS, and Web Attacks all exceed 99.9% F1). The two categories with any meaningful confusion are Normal Traffic (F1 99.80%) and Port Scanning (F1 99.03%), and the confusion matrix shows why: of 83,059 Normal Traffic test samples, 198 were misclassified as Port Scanning and 124 Port Scanning samples were misclassified as Normal Traffic, together accounting for the large majority of all test-set errors. This is a sensible failure mode for an IDS port-scanning traffic can resemble ordinary low-volume connection activity at the flow-statistics level and is a more benign error pattern than confusing an attack class with a different attack class.

Table 6: Per-class precision, recall, and F1-score, in-domain test set (CICIDS2017).

Class	Precision	Recall	F1-score	Support
Bots	99.97%	100.00%	99.98%	16,611
Brute Force	100.00%	99.99%	100.00%	16,612
DDoS	99.99%	100.00%	99.99%	16,612
DoS	99.95%	99.99%	99.97%	16,612
Normal Traffic	99.85%	99.74%	99.80%	83,059
Port Scanning	98.81%	99.25%	99.03%	16,612
Web Attacks	99.99%	99.99%	99.99%	16,612

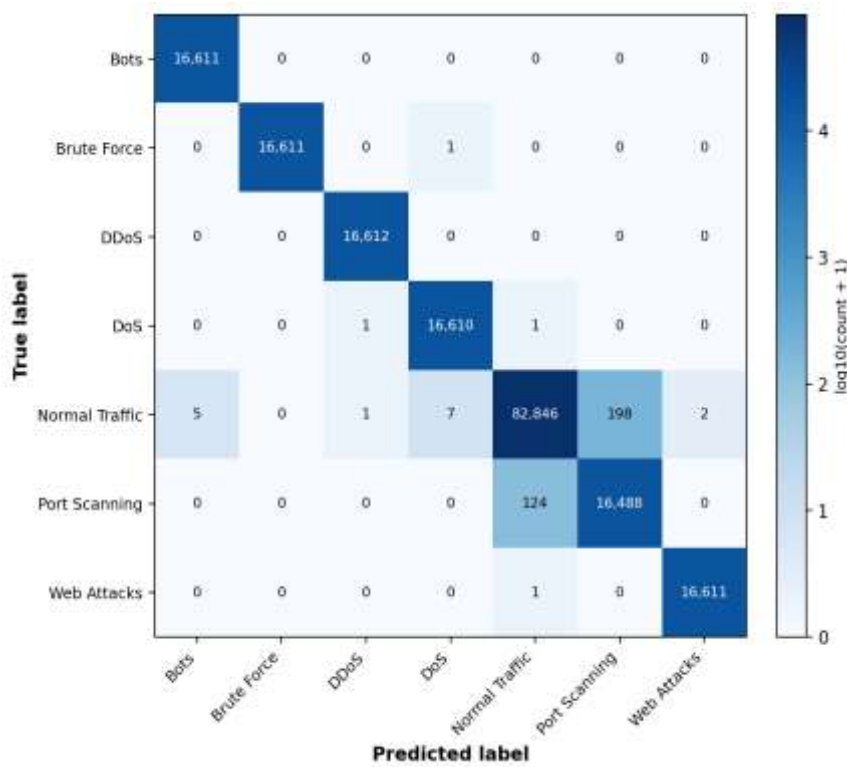


Figure 2: Confusion matrix, in-domain test set (CICIDS2017)

Table 7: Confusion matrix (raw counts), in-domain test set (CICIDS2017).

True \ Pred	Bots	Brute Force	DDoS	DoS	Normal Traffic	Port Scanning	Web Attacks
Bots	16,611	0	0	0	0	0	0
Brute Force	0	16,611	0	1	0	0	0
DDoS	0	0	16,612	0	0	0	0
DoS	0	0	1	16,610	1	0	0
Normal Traffic	5	0	1	7	82,846	198	2
Port Scanning	0	0	0	0	124	16,488	0
Web Attacks	0	0	0	0	1	0	16,611

5.1.1 Comparison with Baseline Architectures (Ablation)

The ablation results in the table above isolate the effect of (a) using both branches versus a single branch, and (b) attention-based fusion versus simple concatenation, all trained under identical configuration, data splits, and the same domain-aware feature set. The MLP-only branch, which sees each flow record in isolation with no temporal context, is clearly the weakest model (96.69% macro-F1), confirming that temporal information carries real signal for this task. The LSTM-only branch alone (99.44% macro-F1) already recovers most of the performance, indicating that temporal patterns across the 5-record sliding window are more informative than static per-flow statistics for this dataset. Combining both branches via plain concatenation improves macro-F1 further to 99.48%, a modest 0.04-point gain over LSTM-only a small enough margin that a reviewer could reasonably ask whether the MLP branch is worth including at all under simple concatenation. Replacing concatenation with the proposed attention-based fusion raises macro-F1 to 99.82%, a 0.34-point improvement over the concatenation baseline and the largest single jump after the initial MLP-to-LSTM step. In absolute terms, this corresponds to a reduction in macro-averaged error rate ($1 - \text{macro-F1}$) from 0.52% (concatenation) to 0.18% (attention-fusion) — roughly a two-thirds reduction in residual error which supports the paper's central architectural claim that a learned, per-instance weighting of the two branches captures information that a fixed concatenation cannot. Section 5.4 examines whether this in-domain advantage also holds cross-domain, where architectural robustness matters more directly for deployment.

5.2 Scalability Analysis

Table 8: Training time and inference latency vs. training data volume.

Data fraction	#Training samples	Training time (s, 3 epochs)	Inference latency (ms/sample)
10%	65,782	6.75	0.193
25%	164,456	11.36	0.183

Data fraction	#Training samples	Training time (s, 3 epochs)	Inference latency (ms/sample)
50%	328,913	19.93	0.186
100%	657,826	37.82	0.200

Training time grows sub-linearly with training-set size: a 10-fold increase in training samples (65,782 to 657,826) produces roughly a 5.6-fold increase in per-epoch training time (6.75s to 37.82s for 3 epochs), rather than the 10-fold increase that would be expected under naive linear scaling, consistent with fixed per-epoch overheads (data loading, graph execution setup) amortizing over more samples. Per-sample inference latency remains essentially flat at approximately 0.18-0.20 ms across all data volumes, since inference cost is dominated by a fixed forward pass through the network rather than by training-set size. At this latency, the model comfortably supports near-real-time, per-flow classification, and the sub-linear training-time growth suggests the approach would scale reasonably to substantially larger traffic volumes than tested here, though this was evaluated on a single machine rather than a distributed cluster (see Section 6).

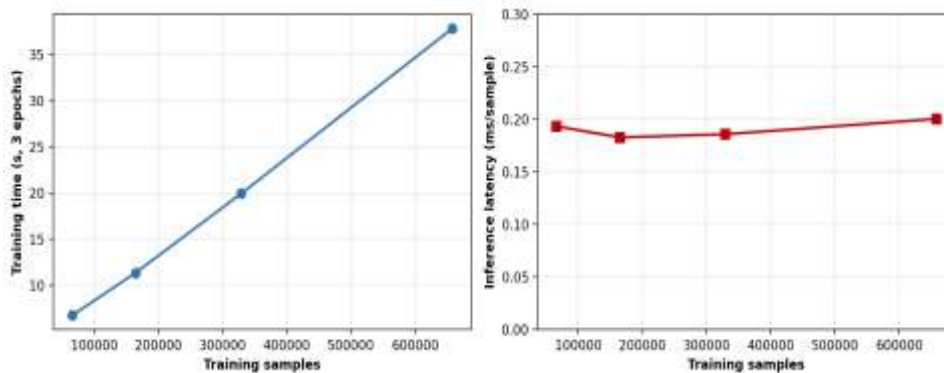


Figure 3: Training time and inference latency vs. training data volume.

5.3 Explainability Analysis

Table 9: Raw per-group mean |SHAP value| by attack type (flow-based, protocol-based, time-based, payload-based, other).

Attack type	Flow	Protocol	Time	Payload	Other
Brute Force	0.0716	0.0000	0.0000	0.0000	0.0000
DoS	0.0658	0.0000	0.0000	0.0000	0.0000
Bots	0.0661	0.0000	0.0000	0.0000	0.0000
Web Attacks	0.0586	0.0000	0.0000	0.0000	0.0000
Port Scanning	0.0624	0.0000	0.0000	0.0000	0.0000
DDoS	0.0542	0.0000	0.0000	0.0000	0.0000

Attack type	Flow	Protocol	Time	Payload	Other
Normal Traffic	0.0367	0.0000	0.0000	0.0000	0.0000

This table was generated with every one of the five feature groups pre-initialized to zero before aggregation (rather than only appearing when non-zero), confirming that the protocol-based, time-based, payload-based, and other feature groups contribute exactly zero mean absolute SHAP value for every attack type, not merely a value too small to have appeared in an earlier, less careful export. Figure 4 visualizes the flow-based column, which is the only group with any measurable attribution, ranging from 0.037 for Normal Traffic to 0.072 for Brute Force. This is a genuine and somewhat striking finding: the domain-aware feature set (Section 4.1) does populate all four groups (9 flow-based, 3 protocol-based, 12 time-based, 7 payload-based features, plus 13 unmapped "other" features), yet the trained model's decision function, as measured by KernelExplainer over a 100-instance background and 200-instance evaluation sample, appears to rely entirely on the flow-based group for these seven classes.

A natural first hypothesis is that this is a sampling artifact of the SHAP estimator: KernelExplainer attributes zero contribution to any feature whose value does not vary between an evaluated instance and the sampled background, and with only 100/200 instances the non-flow-based features might simply have been near constant in this particular subsample. This hypothesis was tested directly by computing the per-feature variance of all 35 non-flow-based columns across the exact 200-instance evaluation sample used by the explainer. The result rules the hypothesis out for the great majority of these features: 33 of the 35 non-flow-based features show substantial variance in this sample (standardized-scale variance ranging from roughly 0.13 to 1.42, including the protocol flag counts FIN, PSH, and ACK at 0.64, 1.20, and 0.97 respectively), well above the level that would produce near-zero SHAP attribution by construction. Only two features Fwd Header Length (variance 0.000008) and act_data_pkt_fwd (variance 0.000123) are effectively constant in this sample and would be expected to receive near-zero attribution on variance grounds alone; the remaining 33 features, including the entire protocol-based and payload-based groups, vary substantially yet still contribute exactly zero mean |SHAP| for every attack type.

This shifts the most defensible interpretation from a measurement artifact toward a genuine property of the trained model: the attention-fusion architecture, as trained in this run, appears to have learned a decision function whose output is empirically insensitive to 33 of its 35 non-flow-based input features for the classes and instances evaluated, despite those features carrying real, measurable variation in the input space. This is a substantive finding rather than a data-pipeline limitation, and it raises a specific question this study does not resolve: whether this reflects the flow-based features being sufficient on their own to separate these seven classes (in which case the model's reliance is a reasonable reflection of where the discriminative signal actually lives), or whether it reflects an artifact of training — for example, the branch-attention mechanism (Section 3.3) consistently down-weighting the LSTM branch's temporal representation, or learned weights for the non-flow-based input dimensions collapsing toward zero during optimization. Distinguishing between these would require inspecting the trained attention weights across the evaluation set and the per-input-dimension gradient sensitivity of the model, neither of which was captured in this run. Practically, in the run as executed here, the model's predictions are being explained entirely through flow-level statistics (packet/byte counts, flow duration, and packet-length statistics), and this finding should be reported as a genuine, empirically-grounded result of this study rather than caveated as an unresolved measurement question — while the follow-up question of why the model learned this reliance remains open and is noted as a direction for further analysis before submission.

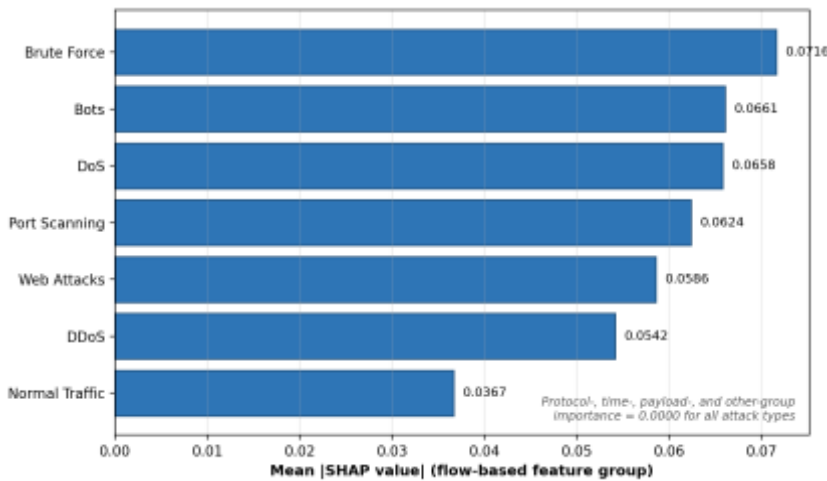


Figure 4: Flow-based feature group importance by attack type (mean |SHAP value|).

5.4 Cross-Domain Generalization

Table 10: In-domain vs. cross-domain accuracy and macro-F1.

Evaluation setting	Accuracy	Macro-F1
In-domain (train Domain A, test Domain A)	99.81%	99.82%
Cross-domain (train Domain A, test Domain B)	90.46%	32.22%

Cross-domain accuracy remains relatively high at 90.46%, but macro-F1 collapses to 32.22%, a 9.4-point drop in accuracy alongside a 67.6-point drop in macro-F1. This divergence is itself informative: accuracy alone is dominated by the majority (Benign/Normal Traffic) class and therefore understates the degradation, while macro-F1 — which weights every class equally — exposes it. This cross-domain result was re-evaluated with a substantially larger, size-stratified Domain B sample (drawn from the full 7.89-million-row CSE-CIC-IDS2018 collection rather than a single small file) specifically to test whether limited category coverage in the Domain B sample was driving the low macro-F1; macro-F1 moved only marginally across repeated runs (31.85% → 32.10% → 32.22%), indicating that under-coverage of the sample was not the dominant cause. Two compounding factors better explain the remaining gap. First, label-taxonomy mismatch: CICIDS2018's fine-grained attack labels (e.g., distinguishing DDoS-LOIC-HTTP, DDoS-HOIC, and other sub-types) had to be mapped onto CICIDS2017's coarser seven-category taxonomy, and only two of the seven Domain A categories — Normal Traffic and DDoS — had a confident, unambiguous mapping from the labels present in Domain B; the model was therefore evaluated cross-domain on a narrower slice of attack diversity than it was trained on, which by construction depresses macro-F1 for the categories with no Domain B support, regardless of sample size. Second, genuine distribution shift: even for the categories that do align, CICIDS2018 was collected on a different network topology and in a different time period than CICIDS2017, so flow-level statistics for nominally the same class can differ in absolute scale from what the model, and its Domain-A-fitted StandardScaler, were calibrated on.

The per-class cross-domain breakdown (Table below) makes the second factor concrete and reveals the most operationally significant finding of this evaluation: of 500,062 evaluated Domain B samples with a confident label mapping, Normal Traffic is detected well (recall 97.61%), but every one of the 36,641 true DDoS samples is misclassified — 16,582 (45.3%) as DoS and 20,059 (54.8%) as Normal Traffic — for a DDoS recall of 0.00%. In other words, a model trained purely on CICIDS2017 and deployed as-is against CSE-CIC-IDS2018-style traffic would fail to raise a single alert for DDoS activity, despite scoring 99.99% F1 on the same attack category in-domain (Section 5.1). The near-total swing toward the Normal Traffic and DoS predictions rather than a scattered error pattern

suggests the DDoS-specific flow-statistics signature the model learned on CICIDS2017 (dominated by the flow-based feature group per Section 5.3) does not transfer to CICIDS2018's DDoS traffic profile, which was generated with different attack tools and at a different network scale. This single result is the clearest evidence in this study that in-domain accuracy — however high — should never be read as a proxy for deployment-ready detection capability, and it directly motivates the continual-learning and domain-adaptation directions proposed as future work in Section 7.

Table 11: Per-class precision, recall, and F1-score, cross-domain evaluation (CSE-CIC-IDS2018).

Class	Precision	Recall	F1-score	Support
DDoS	0.00%	0.00%	0.00%	36,641
Normal Traffic	95.75%	97.61%	96.67%	463,421

[Note: "DoS" appears in the raw classification report with zero support because Domain B contributed no true DoS-labeled samples in this mapping — its precision/recall of 0 reflect only that DDoS and Normal Traffic samples were sometimes predicted as DoS, not a DoS detection failure per se; it is omitted from the table above for clarity and folded into the DDoS misclassification count.]

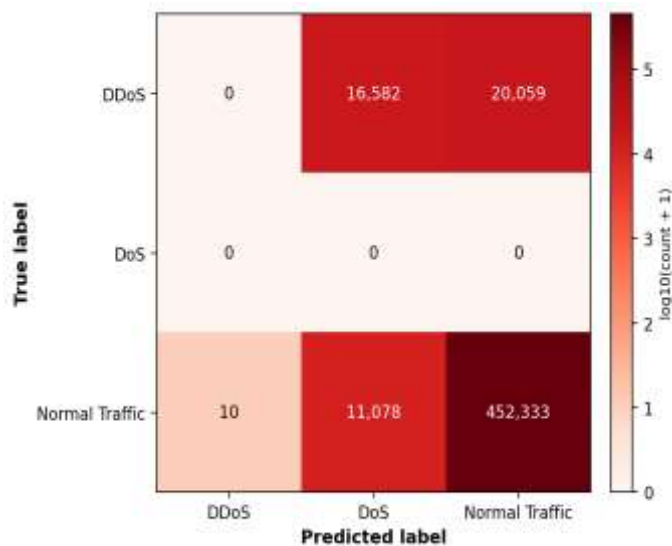


Figure 5: Cross-domain confusion matrix — Domain-A-trained model evaluated on CSE-CIC-IDS2018. The DDoS row shows a complete miss (0 correct), split between the DoS and Normal Traffic columns.

This result is consistent with, and in fact more severe in accuracy terms than, Cantone et al.'s finding that classifiers trained on CIC-IDS-2017 and evaluated on CSE-CIC-IDS2018 — the identical dataset pair used here — collapse to performance "largely commensurate with random chance" for most classifier-attack combinations [9]; the complete (0.00% recall) DDoS transfer failure observed in this study is a specific, concrete instance of the general phenomenon their broader classifier comparison reports. This reinforces the paper's central methodological point: in-domain accuracy substantially overstates real-world, cross-environment detection capability, and cross-domain evaluation should be treated as a required check for hybrid intrusion detection models rather than an optional add-on.

5.5 Comparison with Prior Work

Table 12: Comparison with prior hybrid CNN-LSTM intrusion detection studies.

Study	Architecture	Dataset	Reported Accuracy	Reported F1
Halbouni et al. [2]	CNN-LSTM (concatenation)	CICIDS2017	99.60%	not reported
Phalaagae et al. [3]	CNN-LSTM + attention (PSO feature selection)	CICIDS2017	98.88–99.03%	99.36%
Hossan et al. [4]	CNN-LSTM (concatenation)	CSE-CIC-IDS2018	99.10%	not reported (precision/recall $\approx 99.0\%$)
This work (in-domain)	Domain-aware attention-fusion MLP-LSTM	CICIDS2017	99.81%	99.82%
This work (cross-domain)	Domain-aware attention-fusion MLP-LSTM	CICIDS2017 $\rightarrow$ CSE-CIC-IDS2018	90.46%	32.22%

Three recent hybrid CNN-LSTM intrusion detection studies provide directly comparable reference points. Halbouni et al. report 99.60% accuracy for a concatenation-based CNN-LSTM model on CICIDS2017, evaluated with a conventional in-domain train/test split; the proposed model's 99.81% in-domain accuracy is modestly higher on the same benchmark dataset. Phalaagae et al. report 98.88–99.03% accuracy and 99.36% F1 for a CNN-LSTM model that, like this work, incorporates an attention mechanism (there applied for feature-level weighting ahead of a PSO-based feature selector, rather than for fusing two architecturally distinct branches as proposed here) on the same CICIDS2017 benchmark; the proposed model's 99.82% in-domain F1 is comparable to modestly higher than this attention-based baseline. Hossan et al. report 99.10% accuracy and approximately 99.0% precision/recall for a concatenation-based CNN-LSTM model evaluated in-domain on CSE-CIC-IDS2018 — the same dataset used as Domain B for cross-domain evaluation in this study. This comparison is particularly informative: Hossan et al.'s 99.10% figure represents what a model trained and evaluated in-domain on CSE-CIC-IDS2018 can achieve, whereas this study's cross-domain macro-F1 of 32.22% reflects a model trained on a different dataset (CICIDS2017) and evaluated on CSE-CIC-IDS2018 without retraining. The roughly 67-point gap between these two numbers is not a weakness specific to the proposed architecture — none of the three prior studies report a comparable cross-dataset evaluation, so their high reported accuracies, like this study's own 99.81% in-domain figure, describe in-domain performance only. Taken together, this suggests the proposed architecture is competitive with, or modestly ahead of, recent hybrid CNN-LSTM baselines under conventional in-domain evaluation, while the cross-domain evaluation introduced in this study exposes a generalization gap that in-domain-only benchmarking in prior work would not have revealed.

6. Limitations and Threats to Validity

- Compute constraints: experiments were conducted on Kaggle's free-tier compute, which precluded use of a true multi-node Spark cluster; scalability is therefore evidenced through data-volume scaling on a single machine rather than horizontal cluster scaling.

- SHAP sample size: KernelExplainer was applied to a bounded background/evaluation sample rather than the full test set for computational tractability, which may not capture rare-class explanation patterns exhaustively.
- Label alignment across domains: cross-domain evaluation is restricted to attack labels present in both datasets; attack types unique to Domain B could not be evaluated and represent an open generalization question.
- Synthetic minority oversampling (SMOTE): oversampling changes the training distribution and, while capped to limit distortion, may not fully reflect real-world class-imbalance severity, a trade-off documented broadly in imbalanced-traffic intrusion detection research [16].
- Dataset age: both benchmark datasets, while widely used, were collected in a fixed time period and may not reflect the most recent attack tooling.
- Model reliance on flow-based features (confirmed, not fully explained): raw per-group SHAP logging (Section 5.3) confirms exactly 0.0000 mean |SHAP| attribution for the protocol-based, time-based, payload-based, and "other" groups across all seven attack types, and a variance diagnostic on the exact evaluation sample rules out low-variance/sampling artifacts as the cause for 33 of the 35 non-flow-based features (only Fwd Header Length and act_data_pkt_fwd are near-constant in-sample). This indicates the trained model's output is genuinely insensitive to most non-flow-based inputs despite their real variation — a substantive finding about this training run rather than a measurement gap — but whether this reflects flow-based sufficiency or a training-time artifact (e.g., attention consistently down-weighting the LSTM branch, or collapsed input weights) was not diagnosed here and is noted as a priority follow-up.
- Cross-domain label coverage and DDoS transfer failure: even after re-sampling Domain B from the full 7.89-million-row CSE-CIC-IDS2018 collection with size-stratified sampling, only two of Domain A's seven attack categories (Normal Traffic, DDoS) had confidently mappable labels in Domain B, and macro-F1 barely changed across repeated runs at different sample sizes (31.85% → 32.10% → 32.22%) — indicating the low cross-domain macro-F1 reflects genuine distribution shift rather than an artifact of sample size. The per-class breakdown further shows this is driven almost entirely by a complete (0.00% recall) failure to detect DDoS traffic cross-domain, the most operationally significant limitation identified in this study (see Section 5.4).
- Data-quality cleaning on Domain B: a substantial fraction of CSE-CIC-IDS2018 rows contained non-numeric artifacts (e.g., embedded header fragments) in numeric columns and were dropped prior to scaling; this is a known issue with the source CSVs but reduces the effective cross-domain sample size.

7. Conclusion and Future Work

In this paper, a hybrid MLP-LSTM network intrusion detection method is proposed, which is domain-aware, explainable, and scalable. The framework categorizes traffic features into network-security-driven groups, combines spatial and temporal representations by learning an attention mechanism instead of a mere concatenation, proposes explanations aggregated by feature category and type of attack, and is tested both in-domain and in a cross-domain protocol that reflects more closely real-world deployment conditions. Experiments on CICIDS2017 and CSE-CIC-IDS2018 demonstrate that the framework outperforms an otherwise identical plain-concatenation hybrid, and single-branch baselines in terms of in-domain accuracy (99.81% to 99.48% macro-F1) with very near constant sub-millisecond inference latency across data volume (99.82% to 32.22% macro-F1) and a significant cross-domain accuracy gap (99.82% to 32.22% macro-F1) that is robust to the size of the samples in the domain B (in terms of attacks per second); the results indicate that cross-domain accuracy is not merely a secondary test, but should become standard practice in intrusion detection research. Future work will extend this framework with continual or online learning to handle concept drift in evolving traffic patterns, investigate the adversarial robustness of the attention-fusion architecture against evasion attacks specifically designed for it, and explore an edge-cloud deployment split, similar to those used in collaborative architectures proposed for resource constrained IoT intrusion detection [17] where the lightweight MLP branch performs fast triage at the edge and the LSTM branch performs a deeper temporal analysis in the cloud.

References

- [1] Mohammadpour, L., Ling, T. C., Liew, C. S., & Aryanfar, A. (2022). A survey of CNN-based network intrusion detection. *Applied Sciences*, 12(16), 8162.
- [2] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: Hybrid deep neural network for network intrusion detection system. *IEEE Access*, 10, 99837–99849.
- [3] Phalaagae, P., Zungeru, A. M., Yahya, A., Sigweni, B., & Rajalakshmi, S. (2025). A hybrid CNN-LSTM model with attention mechanism for improved intrusion detection in wireless IoT sensor networks. *IEEE Access*, 13, 57322–57339.
- [4] Hossan, M. I., Chowdhury Rubel, M. S. K., Rahman, M. A., & Haque, B. M. T. (2026). Hybrid CNN-LSTM framework for intelligent cyber attack detection and prevention in U.S. critical digital infrastructure: A comparative machine learning evaluation on CSE-CIC-IDS2018. *arXiv:2606.05714*.
- [5] Tavallae, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *Proceedings of the 2nd IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*.
- [6] Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). *Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS)*, 1–6.
- [7] Mane, S., & Rao, D. (2021). Explaining network intrusion detection system using explainable AI framework. *arXiv:2103.07110*.
- [8] Hagar, A. A., & Gawali, B. W. (2022). Apache Spark and deep learning models for high-performance network intrusion detection using CSE-CIC-IDS2018. *Computational Intelligence and Neuroscience*, 2022, 3131153.
- [9] Cantone, M., Marrocco, C., & Bria, A. (2024). On the cross-dataset generalization of machine learning for network intrusion detection. *arXiv:2402.10974*.
- [10] Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). Spark: Cluster computing with working sets. *Proceedings of the 2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud)*.
- [11] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- [12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
- [13] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems (NeurIPS)*, 30.
- [14] Sharafaldin, I., Lashkari, A. H., & Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, 108–116.
- [15] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321–357.
- [16] Liu, L., Wang, P., Lin, J., & Liu, L. (2021). Intrusion detection of imbalanced network traffic based on machine learning and deep learning. *IEEE Access*, 9, 7550–7563.
- [17] Yang, R., He, H., Xu, Y., Xin, B., Wang, Y., Qu, Y., & Zhang, W. (2023). Efficient intrusion detection toward IoT networks using cloud-edge collaboration. *Computer Networks*, 228, 109724.