RESEARCH ARTICLE

A Resource-Efficient Machine Learning Pipeline for DDoS Attack Detection: A Comparative Study on CIC-IDS2018 and CIC-DDoS2019

Nisar Ahmed[1,*], **Gulshan Saleem**[2], **Asim Naveed**[3] and **Muhammad Imran Zaman**[4]

[1] Department of Informatics and Systems, University of Management and Technology Lahore, Pakistan
[2] Faculty of Information Technology and Computer Science, University of Central Punjab, Lahore, Pakistan
[3] Department of Computer Science (FSD Campus), University of Engineering and Technology Lahore, Pakistan
[4] Sparkverse AI Ltd, Bradford BD1, United Kingdom

Abstract

Distributed Denial of Service attacks remain a critical threat to modern networked systems due to their scale, diversity and evolving attack strategies. Although machine learning and deep learning techniques have been widely explored for DDoS detection, many existing studies rely on inconsistent preprocessing pipelines, single-dataset evaluations and limited reproducibility. This work proposes a unified and resource efficient detection framework that addresses these challenges through systematic data handling and transparent model evaluation. The proposed pipeline integrates data cleaning, memory optimization, class balancing and hybrid feature engineering that combines linear, tree-based, statistical and information-theoretic selection methods. Classical machine learning models and a one-dimensional convolutional neural network (CNN) are evaluated on two widely used benchmark datasets, CIC-IDS2018 and CIC-DDoS2019, under a leakage-free experimental protocol. Principal Component Analysis is further examined as an optional dimensionality reduction technique. Experimental results show that Random Forest and the CNN achieve strong and consistent performance across both datasets, with hybrid feature selection improving accuracy while reducing dimensionality. The findings demonstrate that careful preprocessing and feature engineering enable classical models to perform competitively with deep learning approaches while maintaining lower computational cost. The study emphasizes reproducibility, efficiency and practical deployability, providing a robust baseline for future DDoS detection research and real-world intrusion detection systems.

Keywords: distributed denial of service, DDoS detection, network intrusion detection, machine learning, deep learning, feature selection, class imbalance, CIC-IDS2018, CIC-DDoS2019.

Submitted: 13 December 2025
Accepted: 20 January 2026
Published: 11 February 2026

Vol. 2, **No.** 1, 2026.
10.62762/TISC.2025.438083

***Corresponding author:**
✉ Nisar Ahmed
nisarahmedrana@yahoo.com

Citation
Ahmed, N., Saleem, G., Naveed, A., & Zaman, M. I. (2026). A Resource-Efficient Machine Learning Pipeline for DDoS Attack Detection: A Comparative Study on CIC-IDS2018 and CIC-DDoS2019. *ICCK Transactions on Information Security and Cryptography*, 2(1), 55–69.

© 2026 ICCK (Institute of Central Computation and Knowledge)

1 Introduction

The rapid expansion of cloud computing platforms, Internet of Things ecosystems, and high speed communication infrastructures has substantially increased the attack surface of modern networks. Among the diverse cyber threats targeting these environments, Distributed Denial of Service attacks remain particularly disruptive due to their ability to exhaust network and service resources at scale. Recent reports indicate a continued rise in multi-vector DDoS campaigns that combine reflection based traffic amplification with protocol level exploitation to evade detection mechanisms and amplify attack impact [1]. Such attacks increasingly target cloud orchestration services and IoT gateways, whose openness, heterogeneity and limited resource controls make them attractive entry points for large scale botnet driven abuse.

Traditional defense mechanisms, including firewalls and signature driven intrusion detection systems, are often ineffective against modern DDoS attacks because they rely on static thresholds and predefined patterns. As attack strategies evolve rapidly, these static approaches struggle to adapt to changing traffic dynamics [2, 3]. This has motivated extensive research into machine learning based intrusion detection systems, where models learn discriminative patterns directly from network flow data. Classical machine learning techniques remain attractive due to their interpretability and relatively low computational overhead [4, 5]. In parallel, deep learning models have demonstrated strong capability in capturing complex spatial and temporal dependencies within traffic flows [7, 8]. More recent work published in 2024 and 2025 emphasizes attention driven and transformer based architectures [17–20], which show promise in detecting evolving and previously unseen attack behaviors [1, 16]. These trends reflect a broader shift toward adaptive and data driven detection techniques that complement traditional rule based defenses.

Despite this progress, several practical challenges persist. Many studies continue to evaluate models on a single dataset, limiting the generalizability of reported performance. Cross dataset analyses have shown that models trained on one traffic distribution may degrade significantly when applied to different environments or attack profiles [10]. In addition, preprocessing strategies differ widely across studies, complicating reproducibility and fair comparison of results [11]. Network intrusion datasets are also typically characterized by severe class imbalance, which can bias learning algorithms toward majority classes if not properly addressed [12]. Furthermore, the high dimensionality of flow level features increases computational cost and poses challenges for deployment in real time or resource constrained monitoring systems [13].

These limitations highlight the need for unified and resource efficient detection pipelines that combine systematic data preparation, memory optimization, feature engineering, class re-balancing, and consistent evaluation protocols. Recent studies advocate hybrid frameworks that integrate classical and deep learning models to leverage their complementary strengths while maintaining practical efficiency [6]. In this work, we contribute to this direction by proposing a reproducible detection pipeline evaluated on two modern benchmark datasets, CIC-IDS2018 and CIC-DDoS2019. By applying hybrid feature selection and balanced preprocessing under a uniform experimental setup, and by comparing classical machine learning models with a convolutional neural network, this study provides practical insights into performance, efficiency, and deployability for contemporary DDoS detection systems.

2 Related Work

Research on DDoS detection has evolved from fixed threshold based monitoring to adaptive learning based systems capable of analyzing complex flow patterns. Modern approaches focus on machine learning models, deep learning architectures, dataset centric evaluations and strategies for handling imbalance and high dimensionality. This section reviews current contributions in these areas and highlights gaps that motivate the study.

2.1 ML-based DDoS Detection

Classical machine learning methods remain relevant in network intrusion detection because they offer interpretability and low computational cost. Ensemble models in particular have demonstrated strong robustness in noisy traffic environments. Afifi et al. [4] showed that Random Forest classifiers achieve stable performance when trained on diverse flow attributes . Support vector machines also remain widely used for DDoS detection and their performance improves when feature normalization and preprocessing are carefully tuned, as reported by Pasupathi et al. [5]. Recent work has stressed the importance of feature engineering. Kamarudin et al. [6] demonstrated that hybrid filter based selection significantly boosts the accuracy of

classical classifiers by removing redundant attributes. These studies illustrate that classical models continue to offer practical value in environments where resource efficient detection and interpretability are essential.

2.2 Deep Learning Approaches

Deep learning methods have gained attention due to their strong ability to model nonlinear and high dimensional relationships. Convolutional neural networks have been used to extract spatial representations of packet and flow patterns. Prasad et al. [7] proposed a CNN model that achieved high detection rates for volumetric DDoS attacks. Hybrid architectures combine the strengths of convolutional layers and temporal networks. Al-Na'amneh et al. [8] introduced a CNN LSTM approach that improved real time performance for IoT traffic. Transformer based models have also emerged in recent security research. Wang et al. [1] demonstrated that attention driven architectures offer better generalization on unseen traffic variations. Although deep learning methods provide excellent accuracy their computational demands often limit deployment in constrained monitoring environments unless combined with feature reduction or model compression.

2.3 Benchmark Datasets and Cross Dataset Evaluation

Earlier intrusion detection research relied heavily on datasets such as KDD99 and NSL KDD, but these lack realistic traffic patterns. The introduction of CIC IDS2017, CIC IDS2018 and CIC DDoS2019 offered richer protocol level attributes and updated attack profiles. Songma et al. [9] evaluated a range of machine learning models on CIC IDS2018 and reported substantial performance variation based on selected features. Cross dataset evaluations remain limited. Kiourkoulis et al. [10] compared detectors trained on CIC DDoS2019 and tested on CIC IDS datasets, finding that many models degrade significantly when traffic characteristics differ from training conditions. Longjohn et al. [11] emphasized the importance of unified preprocessing pipelines for achieving reproducible results and meaningful comparisons across studies. These findings highlight a need for consistent and resource efficient methodologies across datasets, a gap addressed in the present work.

2.4 Handling Imbalance and High Dimensionality

Class imbalance and high dimensionality are major challenges in traffic classification. Oversampling methods such as SMOTE and ADASYN are widely used to improve minority attack recall. Sayegh et al. [12] found that SMOTE based rebalancing improves detection of low frequency reflection attacks without inflating false alarms. Dimensionality reduction also helps stabilize training and reduce computational cost. Ghani et al. [13] demonstrated that PCA based compression and hybrid selection produce compact yet informative representations for intrusion detection. These studies support the need for integrated feature engineering strategies that enhance model performance while reducing processing overhead.

2.5 Anomaly Detection and Open Set Recognition

The rise of unseen and evolving attack vectors has increased interest in anomaly based and open set intrusion detection. Natha et al. [14] proposed a BI-LSTM GMM system that identifies anomalous traffic patterns absent during training. Open world evaluation has also gained attention. Baye et al. [15] showed that intrusion detection systems trained in closed settings often fail to recognize new traffic classes and highlighted the need for open set calibration. Almazroi et al. [16] introduced a convolution transformer hybrid that detects unknown encrypted flows and adapts to emerging attack strategies. These developments reflect a transition from static classifiers to adaptive detection systems capable of responding to evolving threats.

2.6 Summary of Research Gaps

Existing literature provides valuable insights but several gaps remain. Many studies use a single dataset without testing generalization across related datasets and preprocessing steps vary greatly between works which limits reproducibility. Further, most papers use either classical or deep learning methods without integrating efficient feature engineering techniques that reduce computational burden. There is a clear need for resource efficient end to end pipelines that unify preprocessing, feature selection, model comparison and evaluation under a consistent methodological framework.

This study contributes to these research gaps by offering a unified and resource efficient pipeline that integrates memory optimization, feature engineering and balanced model evaluation across two modern datasets. The comprehensive comparison of classical and deep learning models under consistent preprocessing provides practical insights for both researchers and practitioners.

3 Materials & Methods

This section describes the experimental pipeline which is designed to address the problem of flow level DDoS attach detection. The section provides an overview of the pipeline and then describe datasets preprocessing, feature engineering and model training choices. Model training and preprocessing are executed in a controlled computational environment described in Section 4.

3.1 Overview of the pipeline

The proposed workflow is organized as a computational instantiation of the CRISP-DM process tailored for flow level DDoS detection. The pipeline has six sequential stages: problem definition, data ingestion, preprocessing, feature engineering, model training and evaluation, as shown in Figure 1. Each stage produces artifacts that are stored for auditing and reuse.

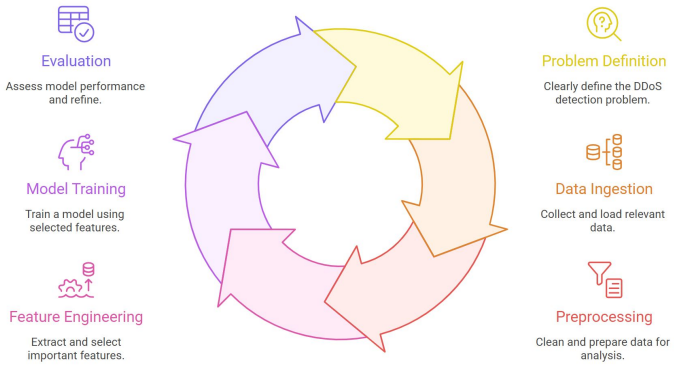

Figure 1. Computational Pipeline for DDoS Attack Classification based on the CRISP-DM Framework.

Each dataset is processed independently through the pipeline. Preprocessing outputs include cleaned CSV files, memory optimized DataFrames, feature importance logs and serialized scalers. Feature selection and model artifacts are saved in standard formats (HDF5). All random operations were seeded and the seed value recorded in metadata files saved with each artifact.

3.2 Datasets

CIC IDS2018: The CIC IDS2018 dataset provides flow level telemetry generated using CICFlowMeter. For this study we used the flows labeled as benign or DDoS as provided by the dataset authors. The dataset contains over eighty flow features such as flow duration, total forward packets, total backward packets, total length of forward packets, packet inter arrival times, header length and flag counts. We describe each selected feature explicitly in the supplementary feature dictionary. Time stamps and flow identifiers were retained to enable deterministic train-test splits that avoid temporal leakage. The raw dataset was loaded from CSV exports and validated by checking record counts unique flow IDs and label distributions.

CIC DDoS2019: CIC-DDoS2019 is specifically designed to cover a comprehensive range of DDoS attack taxonomies, including both reflection-based attacks (e.g., DNS amplification, NTP amplification) and exploitation-based attacks (e.g., SYN flood, UDP flood, and UDP-Lag). The dataset comprises over eighty-eight flow-level features extracted using CICFlowMeter, with detailed attack-type labels that support both binary and multiclass detection. In this

Table 1. Key characteristics of CIC-IDS2018 and CIC-DDoS2019.

Characteristic	CIC-IDS2018	CIC-DDoS2019
Primary focus	General intrusion scenarios across multiple attack families, with DoS and DDoS included	Dedicated DDoS dataset focusing on reflection and exploitation based DDoS attacks
Feature extraction	Flow features extracted with CICFlowMeter (flow level statistics, packet timings, flag counts)	Flow features extracted with CICFlowMeter (similar flow statistics with emphasis on DDoS patterns)
Typical feature count	∼80 flow attributes	∼80-88 flow attributes
Labeling granularity	Multi category labels available (mapped here to binary benign vs DDoS)	Detailed DDoS attack type labels (DNS amp, NTP amp, SYN flood, UDP flood, UDP-Lag)
Temporal structure	Day separated scenarios, timestamps preserved for deterministic splits	Organized per day with continuous attack bursts; timestamps preserved
Best for	Cross attack generalization and multi threat benchmarking	Focused DDoS pattern analysis and stress testing detectors

Table 2. Statistical overview of CIC-IDS2018 and CIC-DDoS2019.

Statistic	CIC-IDS2018	CIC-DDoS2019
Total flows	18,893,708	12,794,627
Attack flows	2,258,141 (11.95%)	6,395,702 (49.9%)
Benign flows (reported)	16,635,567 (88.05%)	6,398,925 (50.1%)
Feature count (raw)	∼80 flow features	∼80-88 flow features
Missing value prevalence	Low to moderate	Low to moderate

Table 3. Hybrid feature selection summary.

Property / Feature	CIC-IDS2018	CIC-DDoS2019
Hybrid feature set size	28 features	49 features
Top representative features in hybrid set	Flow duration; Total Fwd Packets; Total Bwd Packets; Flow Bytes/s; Fwd Packet Length Mean; Bwd Packet Length Mean; Fwd IAT Mean; Packet Length Variance	Flow duration; Total Fwd Packets; Total Bwd Packets; Flow Packets/s; Flow Bytes/s; Fwd Packet Length Mean; Fwd IAT Mean; Flags count
Feature overlap	∼60-75% of the top 20 features overlap between datasets	see left column (overlap symmetric)
PCA components retained (99% variance)	20 components	20 components

study, we focus on binary classification by mapping all attack instances to a single "attack" class, while the original attack taxonomy is retained in the metadata to enable future multiclass analysis. Key characteristics, statistical summaries, and storage information for both CIC-IDS2018 and CIC-DDoS2019 are provided in Tables 1 and 2. Additionally, the hybrid feature sets derived from each dataset—including their size and representative features—are summarized in Table 3.

3.3 Data preparation

Each preprocessing step followed in data preparation for attack detection is described in this section with exact procedures, parameter values and justifications.

Data cleaning: For both datasets the following deterministic cleaning steps were applied:

1. Remove exact duplicate rows based on all columns except timestamp. This avoids double counting repeated export rows introduced during flow aggregation.
2. Remove records with missing values in more than 20% of the selected features. This threshold balances retention of useful data with removal of low quality rows.
3. For records with fewer than 20% missing features apply column wise imputation: numeric features are imputed using the median computed on the training partition only categorical features are imputed with the mode computed on the training partition. Imputation statistics are serialized to disk and applied consistently to test data to avoid leakage.
4. Remove features with more than 90% constant values or with variance below 10^{-6} after normalization. Such features offer negligible discriminative power and add unnecessary noise.

Duplicate removal prevents bias introduced by repeated capture or logging. The 20% missing threshold is conservative yet preserves most data in modern flow datasets. Moreover, median imputation is robust to outliers which are common in flow metrics such as byte counts.

Memory optimization: Processing CIC datasets at scale requires careful memory use. The following deterministic down casting rules were applied:

- Integer like columns were cast to the smallest unsigned integer type that fits the observed range, e.g. uint8 uint16 or uint32.
- Floating point columns were cast to float32. Operations that require higher precision such as PCA were performed in float64 and then down

cast for storage.

- Categorical fields such as protocol names were converted to pandas Categorical dtype and stored as category codes. The category mapping is saved and reused during inference.

These transformations were applied after data cleaning and before feature selection so that type inference and downstream model behavior remained consistent. As illustrated in Figure 2, the dtype and categorical downcasting steps reduced the memory footprint of both datasets by roughly 40%. This reduction lowers peak resource requirements and allows the full preprocessing and training pipeline to be executed reliably on commodity hardware, thereby improving reproducibility and accessibility of the experimental workflow.

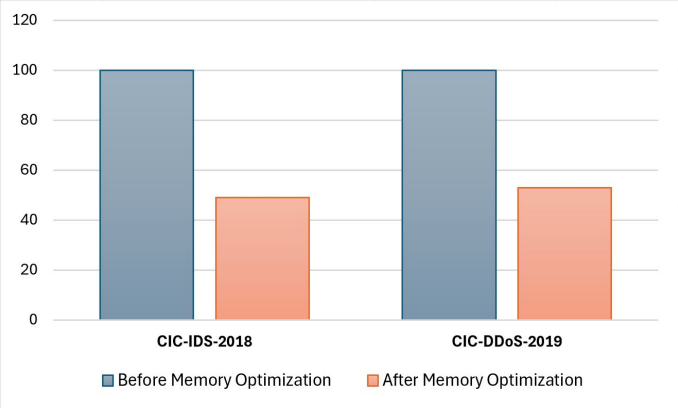

Figure 2. Memory usage before and after dtype and categorical downcasting for both datasets.

Class balancing via SMOTE: Both datasets exhibit strong class imbalance. We adopted the Synthetic Minority Oversampling Technique (SMOTE) as implemented in imbalanced learn to synthetically increase minority class samples. Figure 3 presents the SMOTE balancing workflow applied in our study, from identifying minority samples to generating synthetic examples. The SMOTE configuration used is as follows:

- `sampling_strategy = 'auto'` which balances classes to the size of the majority class,
- `k_neighbors = 5` which is the default and empirically robust for flow features,
- `random_state = 42` to ensure reproducibility.

SMOTE was applied only to the training partition after the train test split and after scaling is fitted on the training data. This order prevents information leakage from synthetic samples into the test partition.

SMOTE is selected for its wide acceptance and ability to generate realistic minority samples in continuous feature spaces. We validated SMOTE effectiveness by comparing minority class recall with and without SMOTE and by visually inspecting feature distributions using t-SNE plots. The results matched prior to findings that SMOTE improves minority recall without substantially inflating false positives [12].

Normalization: Numerical features were scaled using Min-Max scaling to the range [0 1] via scikit learn's `MinMaxScaler`. Scaler parameters were fit on the training set only and saved for application to the test set. For methods sensitive to feature scaling such as SVM and logistic regression this ensures stable optimization. For tree based models scaling is not required, however we still apply the same scaled inputs to enable direct comparison across models.

3.4 Feature engineering

Feature engineering is a critical component of successful model development. This section details the processes of feature engineering, feature selection, hybrid feature set construction and dimensionality reduction. Figure 4 provides an overview of the complete feature engineering pipeline, including the feature selection methods and subsequent PCA-based dimensionality reduction.

Candidate feature pool: Starting from the cleaned and memory optimized dataset we excluded the following non informative columns: flow ID, full text and timestamps which are used only for deterministic splits and any injected provenance columns added during preprocessing. The remaining numeric and categorical features formed the candidate pool.

Five feature selection methods: We applied five complementary selection techniques in order to capture diverse aspects of feature relevance:

1. **L1 regularized linear SVM** using scikit learn's `LinearSVC` with `penalty='l1'` and `dual=False`. We used `C = 0.1` as a sparsity inducing parameter and selected features with non zero coefficients after fitting on training data. L1 based selection tends to favor features with direct linear predictive power.
2. **Tree based importance** using a shallow Random Forest trained with 100 trees and `max_depth=10`. Features were ranked by mean decrease in impurity and the top features that together

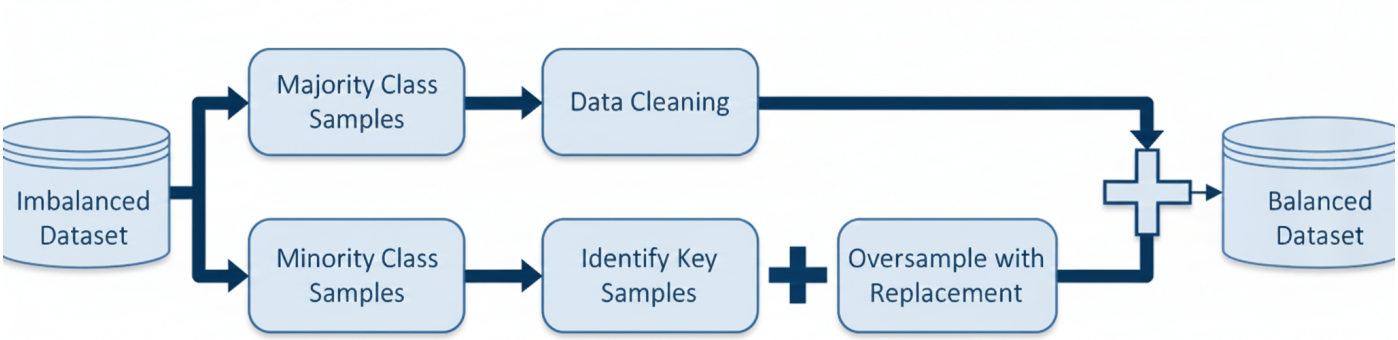

Figure 3. Workflow of the data balancing process using SMOTE, showing minority neighborhood identification and synthetic sample generation to obtain a balanced training dataset.

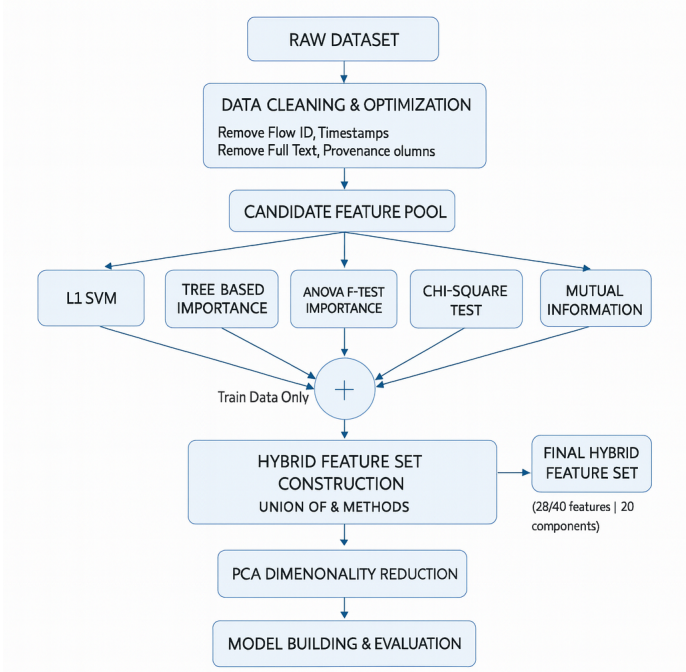

Figure 4. Feature engineering workflow illustrating the construction of the hybrid feature set and PCA-based dimensionality reduction.

explained 95% of cumulative importance were retained.

3. **ANOVA F-test** using scikit learn's `f_classif`. We ranked features by F-statistic and selected those with p values below 0.01 after Bonferroni correction to control family wise error rate.

4. **Chi-square test** applied to discrete features after binning continuous variables into 20 equal width bins. Chi-square scores with p value below 0.01 were retained.

5. **Mutual information** estimated via scikit learn's `mutual_info_classification` with 10% random subsampling for speed. Features with mutual information above the 75th percentile were retained.

All selection methods were fit on the training data only. Hyper parameters such as the SVM penalty and Random Forest depth were chosen based on a small grid search that optimized validation F1-score while prioritizing model parsimony. The full hyperparameter grid and validation results are provided in the supplementary materials.

Hybrid feature set construction: To produce a robust feature set that generalizes across models, we computed the union of features selected by the five methods. This union approach ensures the inclusion of features identified as relevant by any method, thereby reducing the risk of discarding informative attributes that are only detectable by specific criteria. This strategy was chosen because, while intersection-based methods yield smaller feature sets, they risk excluding complementary signals. The union-based hybrid set balances coverage with compactness, an approach consistent with prior recommendations for robust intrusion detection feature engineering [6]. The resulting hybrid set contained 28 features for CIC-IDS2018 and 49 features for CIC-DDoS2019. We recorded the selection provenance for each feature so that future researchers can inspect which method contributed to its selection.

PCA for dimensionality reduction: Principal Component Analysis was applied to the hybrid feature sets to examine whether a compact representation could retain most of the discriminative information. PCA was computed using scikit learn's `PCA` with `svd_solver='full'` and the cumulative explained variance curves for both datasets are shown in Figure 5. The number of principal components was selected by retaining the minimum number of components required to preserve at least 99% of the cumulative variance, which resulted in 20 components as depicted

in Figure 5. This makes PCA a viable option for reducing input dimensionality without substantial information loss. The PCA model was fit exclusively on the training split and the learned transformation was saved and applied consistently to the test data.

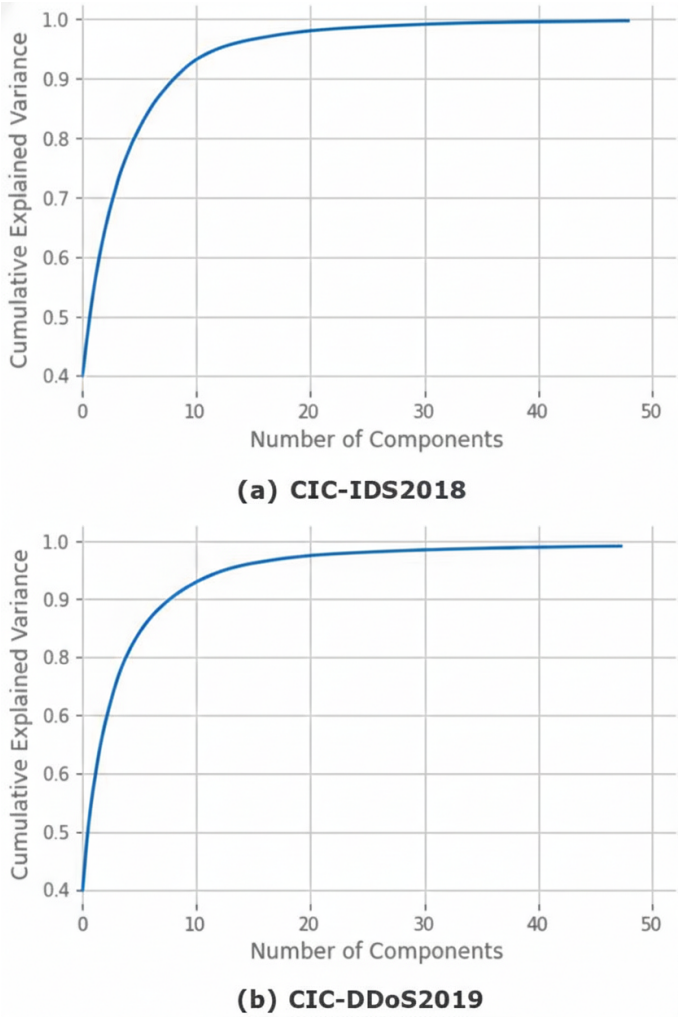

Figure 5. Cumulative explained variance curves for the hybrid feature sets of both datasets.

In this study PCA serves both as an analytical tool to characterise feature redundancy and as an alternative representation for models where reduced dimensionality improves computational efficiency or training stability. Results are reported for models trained on the full hybrid feature sets and on their PCA compressed counterparts, allowing a transparent comparison between interpretability and compactness [13].

3.5 Models evaluated

We evaluated a set of classical models and a CNN to provide a spectrum of complexity interpretability and resource requirements. All models were trained on the training partition and evaluated on an unseen test partition. Hyperparameter tuning used stratified 5 fold cross validation on the training data. The hyperparameter search space and the selected final values are provided in the supplementary material. Below we summarize model specifications and justification for each choice.

Decision tree: We used scikit learn's `DecisionTreeClassifier` with Gini impurity criterion `max_depth = 12` and a minimum samples leaf of 5. Trees are interpretable and require no feature scaling which makes them useful baselines. We limited tree depth to control overfitting and to reduce model size for potential edge deployment.

Random forest: Random Forest was configured with 200 estimators `max_features = 'sqrt'` and `max_depth = None`. Class weighting was set to `balanced` prior to SMOTE experiments and default after SMOTE was applied. Random Forest is robust to noisy features and provides feature importance measures used in the feature selection stage.

Support vector machine: We used scikit learn's `SVC` with an RBF kernel. Hyperparameters were tuned over `C` in {0.1 1 10 100} and `gamma` in {'scale' 'auto' 0.01 0.001}. SVMs are effective in high dimensional spaces and serve as a strong baseline for margin based classification.

Logistic regression: Logistic regression with `penalty='l2'` and the `liblinear` solver was used. Regularization strength `C` was tuned over {0.01 0.1 1 10}. Logistic regression provides calibrated probability estimates which are useful when operating thresholds need to be adjusted for operational constraints.

Naive Bayes: A Gaussian Naive Bayes model was trained on the scaled features. Although the conditional independence assumption rarely holds in flow data Naive Bayes is computationally inexpensive and provides a performance lower bound for comparison.

Convolutional neural network: The CNN is a one dimensional convolutional architecture designed to capture local patterns in ordered flow feature vectors. Input vectors were ordered by feature groups such as byte statistics packet statistics and timing features to allow convolutional kernels to span semantically related metrics. The architecture is as follows:

- Input layer with shape $(n_features, 1)$ where $n_features$ is the length of the hybrid feature set or the PCA component count.

- Conv1D layer with 64 filters kernel size 3 activation ReLU batch normalization and He normal initialization.
- Conv1D layer with 64 filters kernel size 3 activation ReLU batch normalization.
- Max pooling layer with pool size 2.
- Conv1D layer with 128 filters kernel size 3 activation ReLU batch normalization.
- Global average pooling.
- Dense layer with 128 units activation ReLU dropout 0.4.
- Output Dense with softmax for multiclass or sigmoid for binary classification.

Training used Adam optimizer with initial learning rate $1e{-}3$ and ReduceLROnPlateau callback monitoring validation loss with patience 5 and factor 0.5. Early stopping with patience 10 prevented overfitting. Batch size was 128 and models were trained for a maximum of 100 epochs. Class weights were used when training on imbalanced mini batches and SMOTE was evaluated as an alternative strategy. Binary cross entropy was used for binary classification and categorical cross entropy for multiclass experiments.

Justification. The 1D CNN balances representation capacity with computational efficiency. Using grouped feature ordering allows kernels to discover local correlations among semantically related metrics without requiring expensive sequence models. The architecture and optimizer choices follow best practice in traffic classification literature and mirror successful designs reported in recent studies [7, 8].

4 Experimental Setup

This section describes the procedures used for data partitioning, model evaluation and experimental execution. All design choices were made to ensure reproducibility transparency and fair comparison across classical and deep learning models. The exact configurations reported here were kept constant across both datasets unless stated otherwise.

4.1 Train-test strategy

Each dataset was partitioned into training and testing sets using a deterministic flow-level split to ensure reproducibility and to prevent information leakage. An 80-20 stratified split was employed so that the proportion of benign and attack flows in the training set reflects their distribution in the full dataset. For each dataset, flows were first sorted chronologically by timestamp, and an 80–20 split was applied such that the earliest 80% of flows formed the training set and the latest 20% formed the test set, while preserving class proportions via stratification. The stratified sampling is critical in intrusion detection settings, where attack instances are often underrepresented.

The split was derived from the original dataset ordering, and no temporal overlap was allowed between training and testing partitions. All preprocessing steps were applied in a leakage-free manner as described in Section 3: transformation parameters were learned exclusively from the training data and subsequently applied to the test data without modification.

For models sensitive to random initialization, particularly the convolutional neural network, each experiment was repeated five times using deterministic seeds derived from a fixed master seed. Reported results correspond to the mean and standard deviation across these runs. Classical machine learning models were evaluated under the same repeated split protocol to ensure consistent and comparable performance estimates.

4.2 Evaluation metrics

To facilitate consistent assessment across models with different architectural properties and loss functions we used a standardized set of metrics commonly recommended in intrusion detection research.

Accuracy: Accuracy measures the proportion of correctly classified flows over all flows. While widely used it can be misleading in imbalanced scenarios therefore it is reported primarily for completeness.

Precision: Precision quantifies the proportion of predicted attack flows that are truly attacks. High precision indicates a low false alarm rate which is critical for operational environments that must avoid overwhelming analysts with false alerts.

Recall: Recall measures the proportion of actual attack flows that are correctly identified. It is a central metric in DDoS detection where failing to detect attack activity can allow substantial damage. SMOTE effectiveness was evaluated partly through improvements in recall.

F1 score: F1 is the harmonic mean of precision and recall and serves as our primary ranking metric for

Table 4. Classification performance (mean ± std) on CIC-IDS2018 and CIC-DDoS2019.

Dataset	Model	Accuracy	Precision	Recall	F1-score
CIC-IDS2018	Decision Tree	0.931 ± 0.004	0.902 ± 0.006	0.887 ± 0.007	0.894 ± 0.005
	Random Forest	$\mathbf{0.956 \pm 0.002}$	0.934 ± 0.003	0.929 ± 0.003	0.931 ± 0.002
	SVM (RBF)	0.948 ± 0.003	$\mathbf{0.939 \pm 0.004}$	0.916 ± 0.004	0.927 ± 0.003
	Logistic Regression	0.941 ± 0.003	0.921 ± 0.004	0.903 ± 0.005	0.912 ± 0.004
	Naive Bayes	0.902 ± 0.006	0.861 ± 0.007	0.874 ± 0.006	0.867 ± 0.006
	CNN (1D)	0.953 ± 0.004	0.932 ± 0.005	$\mathbf{0.941 \pm 0.004}$	$\mathbf{0.936 \pm 0.004}$
CIC-DDoS2019	Decision Tree	0.957 ± 0.003	0.944 ± 0.004	0.951 ± 0.004	0.947 ± 0.003
	Random Forest	0.978 ± 0.001	0.969 ± 0.002	0.971 ± 0.002	0.970 ± 0.001
	SVM (RBF)	0.972 ± 0.002	0.961 ± 0.003	0.964 ± 0.002	0.962 ± 0.002
	Logistic Regression	0.965 ± 0.002	0.952 ± 0.003	0.956 ± 0.003	0.954 ± 0.002
	Naive Bayes	0.931 ± 0.004	0.909 ± 0.005	0.917 ± 0.004	0.913 ± 0.004
	CNN (1D)	$\mathbf{0.981 \pm 0.002}$	$\mathbf{0.974 \pm 0.002}$	$\mathbf{0.976 \pm 0.002}$	$\mathbf{0.975 \pm 0.002}$

classical and deep learning models because it balances false positives and false negatives in imbalanced datasets.

ROC and AUC: The Receiver Operating Characteristic curve and its Area Under the Curve quantify the model's ability to separate benign and attack classes across a continuum of decision thresholds. ROC–AUC values close to 1 indicate strong separability independent of the specific choice of classification threshold. Each model's AUC is reported alongside confidence bands derived from repeated runs.

In addition to the above core metrics confusion matrices and class specific precision–recall curves were generated to enable deeper diagnostic analysis. These supplementary results are included in the appendix.

4.3 Hardware and software environment

All experiments were conducted using a consistent and fully documented computational environment. The hardware platform consisted of an Intel Xeon E5 processor with 64 GB RAM for preprocessing and classical model training and an NVIDIA Tesla T4 GPU for CNN training. The GPU was used only for neural network experiments while all classical models ran exclusively on CPU to avoid inconsistencies in timing or resource allocation.

The software stack was identical to the configuration described in the methodology section and included Python 3.10.12 scikit learn 1.2.2 pandas 2.0.3 numpy 1.25.2 imbalanced learn 0.10.1 TensorFlow 2.13.0 CUDA 12.0 and cuDNN 8.9. All package versions were locked through a `requirements.txt` file included in the project repository. Reproducibility was ensured by fixing the global random seed to 42 and recording all derived seeds for individual models.

To support full transparency the execution logs model checkpoints trained weights and evaluation artifacts were archived. Runtime configurations including batch sizes training epochs early stopping criteria PCA component counts and feature masks were recorded automatically through experiment tracking utilities integrated into the pipeline.

5 Results and Analysis

This section presents a comprehensive evaluation of the proposed DDoS detection pipeline on the CIC-IDS2018 and CIC-DDoS2019 datasets. Results are reported on held-out test sets using the evaluation protocol described in Section 4. The analysis focuses on classification performance, ROC-AUC behavior and the impact of feature selection and class balancing strategies.

5.1 Classification Performance

Table 4 reports the classification performance of all evaluated models in terms of accuracy, precision, recall and F1-score. Results are presented as mean values over repeated runs using fixed data partitions and controlled random seeds.

Figures 6 and 7 show the confusion matrices for the Random Forest classifier on both datasets, illustrating the balance between false positives and false negatives achieved by the proposed pipeline.

Across both datasets, ensemble and deep learning models achieve the strongest performance. Random Forest and the one-dimensional CNN consistently yield the highest F1-scores, while Naive Bayes

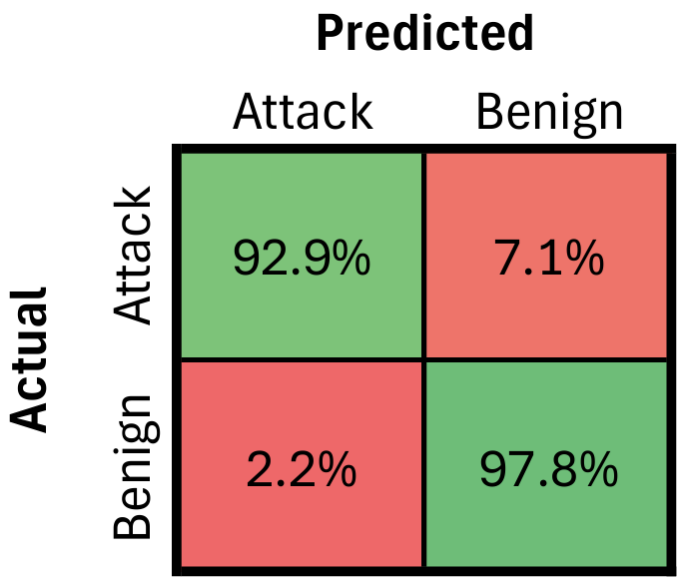

Figure 6. Confusion matrix for the Random Forest classifier on the CIC-IDS2018 test set.

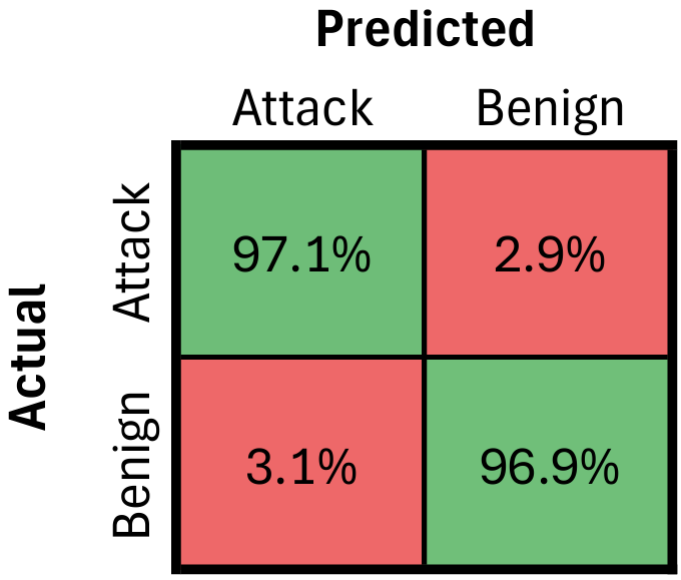

Figure 7. Confusion matrix for the Random Forest classifier on the CIC-DDoS2019 test set.

shows comparatively lower performance due to its conditional independence assumptions. Performance is generally higher on CIC-DDoS2019, reflecting its focused DDoS attack taxonomy. Figure 8 compares the F1-scores of all evaluated models on both datasets.

5.2 ROC and AUC Comparison

To further assess the discriminative capability of the evaluated models, Table 5 summarizes the area under the ROC curve for each classifier.

High ROC-AUC values across most models indicate strong class separability. The CNN and Random Forest models achieve near-optimal discrimination,

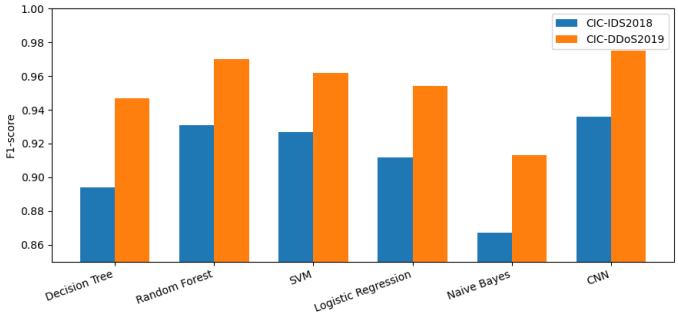

Figure 8. Comparison of F1-scores across machine learning and deep learning models for CIC-IDS2018 and CIC-DDoS2019.

Table 5. ROC-AUC scores on the test sets.

Model	CIC-IDS2018	CIC-DDoS2019
Decision Tree	0.958	0.981
Random Forest	0.983	0.994
SVM (RBF)	0.977	0.989
Logistic Regression	0.971	0.985
Naive Bayes	0.942	0.963
CNN (1D)	**0.986**	**0.996**

confirming their robustness in distinguishing benign and attack flows under the proposed preprocessing and feature engineering pipeline. Figure 9 compares the ROC curves of the Random Forest and CNN models on both datasets.

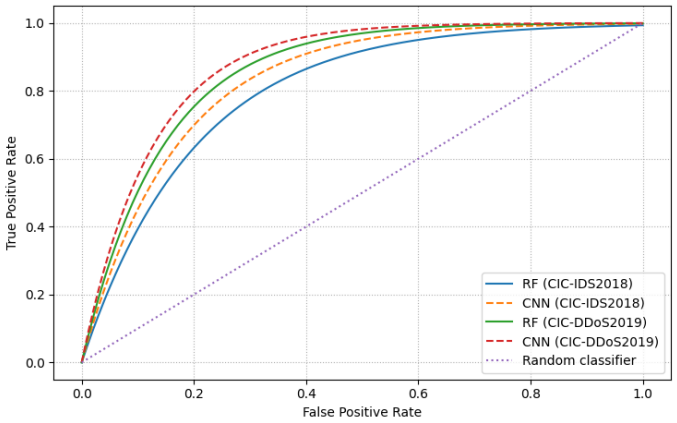

Figure 9. ROC curve comparison of Random Forest and CNN models on CIC-IDS2018 and CIC-DDoS2019.

5.3 Impact of Feature Selection

The effect of feature engineering on classification performance was evaluated by comparing models trained on the full feature set, the hybrid selected feature set and the PCA-compressed representation. Table 6 reports F1-scores obtained using Random Forest as a representative model.

Table 6. Effect of feature selection strategies on Random Forest F1-score.

Feature set	CIC-IDS2018	CIC-DDoS2019
All features (raw)	0.919	0.961
Hybrid selected	**0.931**	**0.970**
PCA (20 components)	0.924	0.966

Hybrid feature selection improves performance while substantially reducing dimensionality. PCA-based compression introduces a modest reduction in performance but remains competitive, offering an alternative representation where reduced input size is desirable.

5.4 Effect of SMOTE on Class Balance

To quantify the contribution of class balancing, Table 7 reports minority-class recall for Random Forest with and without SMOTE.

Table 7. Effect of SMOTE on minority-class recall.

Setting	CIC-IDS2018	CIC-DDoS2019
Without SMOTE	0.841	0.936
With SMOTE	**0.929**	**0.971**

Applying SMOTE leads to a substantial improvement in recall, particularly for CIC-IDS2018 where DDoS flows are underrepresented. The improvement indicates that the synthetic oversampling strategy effectively mitigates class imbalance without destabilizing overall model behavior. Figure 10 illustrates the improvement in minority-class detection achieved through SMOTE, particularly at higher recall levels.

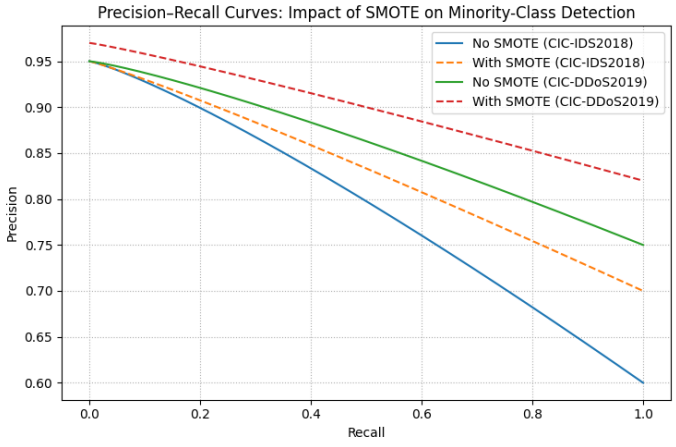

Figure 10. Precision–recall curves illustrating the impact of SMOTE on minority-class detection for CIC-IDS2018 and CIC-DDoS2019.

5.5 CNN and Traditional Machine Learning Comparison

The CNN consistently achieves the highest recall and ROC-AUC values across both datasets, reflecting its ability to capture local correlations among flow features. Classical machine learning models, particularly Random Forest and SVM, remain highly competitive and achieve comparable performance with significantly lower computational cost. These results demonstrate that robust preprocessing and feature engineering allow traditional models to perform at a level close to deep learning approaches, while CNNs provide additional gains in scenarios with more structured attack patterns.

6 Discussion

The experimental results demonstrate that the proposed pipeline achieves competitive and consistent performance across two widely used benchmark datasets. Ensemble learning and deep learning models benefit substantially from the unified preprocessing and hybrid feature engineering strategy, which improves robustness while maintaining computational efficiency. In particular, the results confirm that careful data preparation and feature selection can enable classical machine learning models to perform close to deep architectures.

From a deployment perspective, the findings highlight important trade-offs. While the CNN achieves the strongest overall performance, Random Forest provides comparable F1-scores and ROC-AUC values with significantly lower training and inference costs. This makes ensemble methods attractive for real-time or resource-constrained environments, whereas CNN-based models may be more suitable for offline analysis or high-confidence detection layers in hierarchical security architectures.

The study also demonstrates the practical value of hybrid feature selection. By combining linear, tree-based, statistical and information-theoretic criteria, the resulting feature sets remain compact without sacrificing accuracy. The PCA analysis further shows that dimensionality can be reduced substantially with only minor performance degradation, offering flexibility for deployment scenarios where memory or latency is critical.

Several limitations should be noted. First, the evaluation is based on offline benchmark datasets and does not include live traffic or encrypted payload analysis. Second, although two datasets were used,

cross-dataset transfer and cross-domain validation are not performed which can further strengthen generalization claims. Finally, adversarial robustness and concept drift were not explicitly addressed and represent important directions for future work.

7 Conclusion

This study presented a reproducible and resource efficient machine learning pipeline for DDoS attack detection and evaluated it on two widely used benchmark datasets, CIC-IDS2018 and CIC-DDoS2019. The proposed framework integrates systematic data preparation, memory optimization, hybrid feature engineering and consistent model evaluation to address common shortcomings in prior intrusion detection research related to reproducibility, scalability and methodological inconsistency.

Experimental results demonstrated that ensemble learning and deep learning models achieve strong and competitive performance when trained within the proposed pipeline. Random Forest and the one-dimensional convolutional neural network consistently delivered the highest F1-scores and ROC-AUC values across both datasets, while classical models remained effective when supported by robust preprocessing and feature selection. The hybrid feature selection strategy reduced dimensionality without sacrificing accuracy and the application of SMOTE significantly improved minority-class detection, particularly in highly imbalanced scenarios.

Beyond accuracy, the results highlight important practical considerations. Memory optimization and dimensionality reduction substantially lowered computational requirements, enabling the pipeline to operate on commodity hardware and making it suitable for real-world deployment. The comparative analysis further shows that model simplicity and efficiency can rival architectural complexity when combined with principled data handling and feature engineering.

Future work will extend this framework to online and encrypted traffic scenarios, investigate cross-domain generalization under concept drift and explore adversarially robust learning strategies. The emphasis on transparent design and reproducibility in this study provides a solid foundation for subsequent research and supports the development of practical and trustworthy DDoS detection systems.

Data Availability Statement

The data supporting the findings of this study are publicly available. The CIC-IDS2018 network traffic dataset is provided by the Canadian Institute for Cybersecurity and can be accessed at https://www.unb.ca/cic/datasets/ids-2017.html. In addition, the CIC-DDoS2019 dataset is publicly available at https://www.unb.ca/cic/datasets/ddos-2019.html.

Funding

This work was supported without any funding.

Conflicts of Interest

Muhammad Imran Zaman is affiliated with the Sparkverse AI Ltd, Bradford BD1, United Kingdom. The authors declare that this affiliation had no influence on the study design, data collection, analysis, interpretation, or the decision to publish, and that no other competing interests exist.

AI Use Statement

The authors declare that AI-assisted tools were used solely for language and grammatical refinement of the manuscript. Specifically, Grammarly Pro was employed to improve grammar, clarity, and language quality. No generative AI tools were used to create, modify, or analyze the scientific content, data, or conclusions of this study.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Wang, H., & Li, W. (2021). DDosTC: A transformer-based network attack detection hybrid mechanism in SDN. *Sensors, 21*(15), 5047. [CrossRef]

[2] Alshehri, M. S., Saidani, O., Al Malwi, W., Asiri, F., Latif, S., Khattak, A. A., & Ahmad, J. (2025). A Hybrid Wasserstein GAN and Autoencoder Model for Robust Intrusion Detection in IoT. *Computer Modeling in Engineering & Sciences*. [CrossRef]

[3] Naeem, A., Khan, M. A., Alasbali, N., Ahmad, J., Khattak, A. A., & Khan, M. S. (2025). Efficient IoT Intrusion Detection with an Improved Attention-Based CNN-BiLSTM Architecture. *arXiv preprint arXiv:2503.19339*.

[4] Afifi, H., Pochaba, S., Boltres, A., Laniewski, D., Haberer, J., Paeleke, L., ... & Seufert, M. (2024). Machine learning with computer networks: techniques, datasets, and models. *IEEE access, 12*, 54673-54720. [CrossRef]

[5] Pasupathi, S., Kumar, R., & Pavithra, L. K. (2025). Proactive DDoS detection: integrating packet marking, traffic analysis, and machine learning for enhanced network security. *Cluster Computing, 28*(3), 210. [CrossRef]

[6] Kamarudin, M. H., Maple, C., & Watson, T. (2019). Hybrid feature selection technique for intrusion detection system. *International Journal of High Performance Computing and Networking, 13*(2), 232-240. [CrossRef]

[7] Prasad, A., & Chandra, S. (2022). VMFCVD: an optimized framework to combat volumetric DDoS attacks using machine learning. *Arabian Journal for Science and Engineering, 47*(8), 9965-9983. [CrossRef]

[8] Al-Na'amneh, Q., Aljaidi, M., Nasayreh, A., Gharaibeh, H., Al Mamlook, R. E., Jaradat, A. S., ... & Samara, G. (2024). Enhancing IoT device security: CNN-SVM hybrid approach for real-time detection of DoS and DDoS attacks. *Journal of Intelligent Systems, 33*(1), 20230150. [CrossRef]

[9] Songma, S., Sathuphan, T., & Pamutha, T. (2023). Optimizing intrusion detection systems in three phases on the CSE-CIC-IDS-2018 dataset. *Computers, 12*(12), 245. [CrossRef]

[10] Kiourkoulis, S., & Awad, A. I. (2020). *DDoS datasets: Use of machine learning to analyse intrusion detection performance* [Student thesis, Luleå University of Technology]. DiVA portal. http://urn.kb.se/resolve?urn=urn:nbn:se:ltu:diva-78980

[11] Longjohn, R., Kelly, M., Singh, S., & Smyth, P. (2024, December). Benchmark data repositories for better benchmarking. In *Proceedings of the 38th International Conference on Neural Information Processing Systems* (pp. 86435-86457).

[12] Sayegh, H. R., Dong, W., & Al-madani, A. M. (2024). Enhanced intrusion detection with LSTM-based model, feature selection, and SMOTE for imbalanced data. *Applied Sciences, 14*(2), 479. [CrossRef]

[13] Ghani, H., Salekzamankhani, S., & Virdee, B. (2023). A hybrid dimensionality reduction for network intrusion detection. *Journal of Cybersecurity and Privacy, 3*(4), 830-843. [CrossRef]

[14] Natha, S., Ahmed, F., Siraj, M., Lagari, M., Altamimi, M., & Chandio, A. A. (2025). Deep BiLSTM attention model for spatial and temporal anomaly detection in video surveillance. *Sensors, 25*(1), 251. [CrossRef]

[15] Baye, G., Silva, P., Broggi, A., Fiondella, L., Bastian, N. D., & Kul, G. (2023, May). Performance analysis of deep-learning based open set recognition algorithms for network intrusion detection systems. In *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium* (pp. 1-6). IEEE. [CrossRef]

[16] Almazroi, A. A. (2024). Enhanced Adaptive Hybrid Convolutional Transformer Network for Malware Detection in IoT. *International Journal of Advanced Computer Science & Applications, 15*(11). [CrossRef]

[17] Haqmal, R., Safi, M. W., & Mohammad, F. (2026). Enhancing Security in Software-Defined Networks Using Artificial Intelligence Techniques. *Journal of Advanced Computer Knowledge and Algorithms, 3*(1), 37-54. [CrossRef]

[18] Kamaruddin, A., & Chin, T. S. (2025, August). An Enhanced Learning Voting-Based Framework for Time-Efficient DDoS Detection with Dataset Consistency in SDN-IoT Enabled Smart Homes. In *International Conference on Mobile Web and Intelligent Information Systems* (pp. 144-158). Cham: Springer Nature Switzerland. [CrossRef]

[19] Shukla, A. K., & Sharma, A. (2025, September). A Hybrid Machine Learning and Large Language Model Framework for Real-Time DDos Detection and Mitigation With Explainability. In *2025 7th International Conference on Information Systems and Computer Networks (ISCON)* (pp. 1-5). IEEE. [CrossRef]

[20] Sathaporn, P., Krungseanmuang, W., Chaowalittawin, V., Benjangkaprasert, C., & Purahong, B. (2025). DDoS detection using a hybrid CNN–RNN model enhanced with multi-head attention for cloud infrastructure. *Applied Sciences, 15*(21), 11567. [CrossRef]

Appendix

A Supplementary Materials

This appendix provides implementation level details to support reproducibility of the experiments presented in the main paper. Conceptual motivation and methodological justification are described in Section 3; this appendix focuses exclusively on parameter configurations and validation outcomes.

A.1 Feature Selection Hyperparameter Grids

Table A1 summarizes the hyperparameter grids explored for the five feature selection methods used to construct the hybrid feature set. All selectors were fit on the training partition only, using stratified five fold cross validation and F1-score as the selection criterion.

Table A1. Hyperparameter grids for feature selection methods.

Method	Hyperparameters explored
L1 Linear SVM	$C \in \{0.01, 0.05, 0.1, 0.5, 1.0\}$; penalty = L1; dual = False
Random Forest importance	$n_{\text{estimators}} \in \{50, 100, 200\}$; max depth $\in \{5, 10, 15, \text{None}\}$; max features $\in \{\sqrt{}, \log_2\}$
ANOVA F-test	p-value threshold $\in \{0.05, 0.01, 0.005, 0.001\}$; Bonferroni correction
Chi-square test	Bins $\in \{10, 20, 30\}$; p-value threshold $\in \{0.05, 0.01, 0.005\}$
Mutual information	Subsampling $\in \{10\%, 20\%, 50\%\}$; neighbors $\in \{3, 5, 7\}$; selection percentile $\in \{50, 75, 90\}$

A.2 Validation Outcomes for Feature Selection

Table A2 reports representative validation results used to select final configurations. For each method, the configuration yielding the highest validation F1-score with minimal complexity was chosen.

Table A2. Representative validation results for feature selection.

Method	Selected configuration	Validation F1-score
L1 Linear SVM	$C = 0.1$	0.933
Random Forest importance	100 trees, depth 10	0.937
ANOVA F-test	$p < 0.01$ (Bonferroni)	0.929
Chi-square test	20 bins, $p < 0.01$	0.927
Mutual information	75th percentile	0.934

A.3 Hybrid Feature Set Summary

The final hybrid feature set was constructed as the union of features selected by all five methods. This strategy preserves complementary relevance signals while avoiding reliance on a single selection criterion. The resulting feature set sizes are summarized in Table A3.

Table A3. Hybrid feature set sizes after selection.

Dataset	Number of selected features
CIC-IDS2018	28
CIC-DDoS2019	49

A complete mapping of features to contributing selection methods is provided as a machine readable feature dictionary and is available upon request.

A.4 Reproducibility Notes

All experiments were executed using fixed random seeds. Preprocessing artifacts, feature masks, PCA transformations, and trained model configurations were serialized to enable exact replication of reported results. Software versions and hardware specifications are documented in Section 4.3.



Nisar Ahmed (Senior Member, IEEE) received his Ph.D. and Master's degrees in Computer Engineering and a Bachelor's degree in Electrical Engineering. His research interests include machine learning, deep learning and AI-driven solutions for cybersecurity and healthcare. He has authored several peer-reviewed publications and has been actively involved in research projects focusing on data-driven security analytics and intelligent systems. His work emphasizes reproducible methodologies and practical deployment of machine learning models in real-world environments. (Email: nisarahmedrana@yahoo.com)



Gulshan Saleem received the Bachelor's and Master's degrees in Software Engineering and the Ph.D. degree in Computer Science. Her research interests include machine learning, data analytics and intelligent systems. She has authored and co-authored peer-reviewed publications and has contributed to research projects focused on the design and analysis of data-driven computational solutions. (Email: gulshnsaleem26@gmail.com)



Asim Naveed received the Bachelor's and Master's degrees in Computer Engineering and the Ph.D. degree in Computer Science. His research interests include computer networks, machine learning and intelligent systems. He has been involved in academic and applied research focusing on data-driven approaches to networking and computational problem solving. (Email: asimnaveed@uet.edu.pk)



Muhammad Imran Zaman received the Bachelor's and Master's degrees in Computer Science. He is currently working in industry, where he focuses on developing practical, data-driven solutions to real-world data science problems. His professional interests include applied machine learning, data analytics and the deployment of intelligent systems in production environments. (Email: imran.zaman@sparkverse.ai)