EAI Endorsed Transactions on AI and Robotics Research Article EAI.EU

Evolutionary Feature Reduction and Edge-Optimized CNN Inference for Large-Scale IoT DDoS Detection

Pravir Chitre[1,*], Premkumar Sivakumar[2]

[1,2]School of Computer Science & Engineering, Galgotias University, Greater Noida (UP), India

Abstract

The rapid proliferation of Internet of Things (IoT) deployments has introduced significant security vulnerabilities, particularly due to Distributed Denial-of-Service (DDoS) attacks launched through compromised IoT botnets. Real-time detection of such attacks at the network edge remains challenging because of high feature dimensionality, severe class imbalance between benign and attack traffic, and strict latency constraints of resource-constrained IoT gateways. This paper aims to design and evaluate a unified framework for large-scale IoT DDoS detection that reduces feature dimensionality, improves classification performance under imbalanced conditions, and enables low-latency edge deployment suitable for real-time gateway environments. The proposed framework employs a multi-phase pipeline integrating evolutionary feature reduction, deep learning classification, and edge-optimized deployment. A Weighted Genetic Algorithm (W-GA) is used to select a compact and importance-ranked subset of discriminative features from the original high-dimensional representation. A one-dimensional Convolutional Neural Network (CNN) is then trained on the feature set with reduced weights to capture the characteristic of local co-occurrence patterns of volumetric DDoS traffic. Finally, the trained model is exported and deployed using ONNX Runtime for efficient inference on IoT gateway hardware. Experimental evaluation on the CIC-IoT-2023 dataset demonstrates that the proposed W-GA+CNN framework consistently outperforms baseline classifiers in terms of classification effectiveness and inference throughput while maintaining sub-millisecond edge inference latency. The proposed evolutionary feature reduction and edge-optimized CNN framework provides an effective and deployment-ready solution for real-time large-scale IoT DDoS detection, making it suitable for practical intrusion detection deployment in production IoT gateway environments.

Received on 07 May 2026; accepted on 17 July 2026; published on 10 August 2026

Keywords: Internet of Things (IoT), Distributed Denial-of-Service (DDoS), Intrusion Detection System (IDS), Genetic Algorithm, Feature Selection, Real-Valued Weight Encoding, Convolutional Neural Network (CNN), Edge Computing, ONNX Runtime, CIC-IoT-2023, Class Imbalance, Matthews Correlation Coefficient (MCC), Tail Latency, IoT Gateway Deployment

doi:10.4108/airo.12940

1. Introduction

The proliferation of Internet of Things (IoT) devices [1] has ushered in a massive influx of network-connected endpoints unknown to the modern world as smart homes, healthcare infrastructures, industrial automation systems and driving. By 2030, the world is estimated to have more than 39 billion active IoT connections [2, 3] an enormous, constantly under-secured attack surface for an attacker. IoT devices are structurally prone to being exploited: most IoT devices are built on microcontroller class processors with 64-256 KB of RAM, have vendor security support for only 2-5 years out of a 10-20-year deployment lifecycle, and often use factory default credentials or unencrypted protocols [4].

These characteristics have facilitated the creation of IoT-based Distributed Denial-of-Service (DDoS) botnets of unprecedented size - the Mirai botnet demonstrated in 2016 that commodity IoT devices could sustain attack volumes of over 1 Tbps [5], a threshold now well surpassed by successor families including Mozi,

*Corresponding author. Email: pravir.chitre@gmail.com

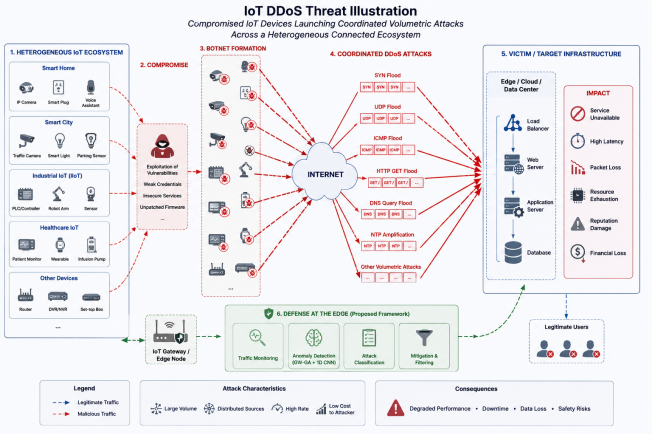

Figure 1. IoT DDoS threat illustration showing compromised IoT devices launching coordinated volumetric attacks across a heterogeneous connected ecosystem

BotenaGo and Fodcha. A IoT dataset of unprecedented diversity UDP flood, TCP SYN flood, HTTP flood, SlowLoris, and four Mirai sub-variants over a 105-device physical testbed, has been released by the Canadian Institute for cybersecurity.

This paper proposes and evaluates a novel framework for the detection of DDoS attacks in this dataset using a combination of evolutionary feature reduction and edge optimized convolutional classifier and systematic Open Neural Network Exchange (ONNX) deployment benchmarking. Four challenges in detecting DDoS attacks in large scale IoT traffic stream in practice, the original detection approaches for DDoS are not sufficient to solve it all together.

First, volumetric scale: In the case of CIC-IoT-2023, a volume of around 46 million flow records has been released, thus the centralised bath classifier is incompatible with real-time gateway detection.

Second, feature dimensionality - standard flow analysis results in 48 features per record imposing onset of computational overhead beyond IoT gateway processing budgets shared with routing and firewall functions [6].

Third is severe class imbalance: the partition used for the CIC-IoT-2023 test has 94.7% attack prevalence, making it difficult to use accuracy as a robust metric, and necessitating the use of measures with sensitivity to minority-class (benign) test performance such as Matthews Correlation Coefficient (MCC), False Alarm Rate [7] as illustrated in Figure 1 [8].

Fourth, latency constraint: IoT gateway deployment requires sub millisecond inference to process flows at line rate a 10,000 flow/sec. gateway needs throughput of over 10,000 classifications/sec. Computationally heavy models cannot sustain, processing backlog [9].

Feature reduction is a prerequisite to scalable IoT intrusion detection deployment. The 48 raw CIC-IoT-2023 features contain large redundancy, the rate-based metrics are highly correlated, the statistics from bidirectional packets are linearly correlated to the duration of the flow, and the protocol-specific flags do not provide any discriminative information for non-TCP attacks [10]. Retaining redundant features increases variance of the decision boundaries without adding discriminative information to them, degrading the generalisation task while increasing the cost of inference.

The Weighted Genetic Algorithm (W-GA) [11–14] used in this part of the work solves this as a direct classification performance optimisation: where different filter m ethods e valuate f eatures i ndividually, W-GA evaluates candidate subsets of features by their actual downstream F1 score. Its real valued weight encoding has the additional result that it gives an importance ordered feature sequence whose positional structure is exploited by the 1D CNN through convolutional receptive fields Figure 2 – a property not available through binary selection methods [10]. The resulting 48-to-5 feature reduction (89.6%) directly allows for the sub-millisecond ONNX edge inference presented in this work that can be implemented in Edge Computing for Real-Time IoT Security. We have experimentally proved that Empirically, this trade-off is highly favourable with maximum degradation of metrics like accuracy, recall and MCC by not more than 1.3% despite the fact that we have discarded 89.6% of features. Thus reducing the computation substantially as the $ModelComplexity \propto O(n \cdot d)$; where n is the number of samples, in this case of CIC-IoT 2023 $\approx$ 46, 000, 000 and d is the number of features which we has been reduced to 5 from 48. This helps in reducing the computation code from 2,20,80,00,000 to 2,30,00,000. Edge computing sees inference being repositioned from centralised cloud servers to devices at the boundary of the network, removing the round-trip latency that cloud based DDoS detection has without being operationally effective. A DDoS attack inside the device subnet needs to be detected at the device's local subnet gateway before it is routed off-site cloud DDoS detection the attack is observed only after the bad traffic has passed through the access network [9]. The Open Neural Network Exchange (ONNX) format allows for this deployment: IoT gateways are not suitable for publishing Python or PyTorch runtimes and low latency hardware-agnostic model executables compatible with C/C++ gateway software stacks are required. ONNX Runtime offers a framework-independent inference (x86, ARM, RISC-V) with automatic SIMD (Single Instruction, Multiple Data) vectorisation, and its graph optimisation (fusion of Conv+ReLU, constant folding) allows to decrease the inference latency by 25-35% compared to the PyTorch

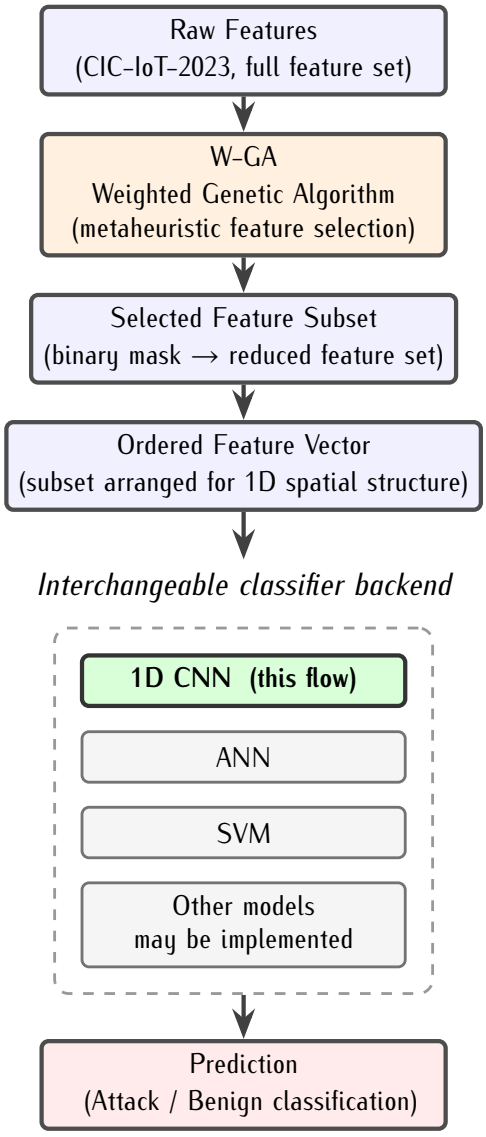

Figure 2. Workflow of the proposed W-GA based feature selection and classifier framework

(native CPU) inference as [15], which directly enables the gateway-viable performance shown in Evaluation Stage 2 of this work. This research aims to achieve the following four objectives:

1. To design and validate a W-GA to reduce 48-featured CIC-IoT-2023 representation to a 5-featured weighted representation that optimises 1D CNN classification performance.

2. To design an edge optimised 1D CNN architecture that leverages importance ordered feature sequence from W-GA to determine local feature co-occurrence patterns that are characteristic of IoT DDoS traffic.

3. To establish a 12-model cohesive evaluation of W-GA+CNN with other models.

To accomplish the above, the original contributions of this paper are four-fold, as follows - firstly, an evolutionary feature reduction algorithm (W-GA), operating at full dataset scale, that converges to 5 features weighted chromosome, i.e. [Flow Duration, Protocol Type, Rate, S-rate, Magnitude], to capture feature selection and importance ranking in one unified representation; secondly, an edge-optimized CNN architecture (1D CNN), which utilizes the weight-ordered feature sequence from W-GA to achieve competitive classification capability with sub-millisecond ONNX inference latency in suitability for IoT Gateway deployment; thirdly, systematic The rest of this paper is organised as follows: Section 2 aims to review the relevant work, Section 3 expresses the CIC-IoT-2023 dataset and data preprocessing processing pipeline, Section 4 presents the proposed W-GA and CNN framework, Section 5 describes the ONNX edge deployment processing workflow, Section 6 and Section 7 specify the experimental setup and evaluation metrics, along with evaluation metrics the results and analysis are discussed. Section 8 provide the discussion and Section 9 conclusions respectively.

2. Related works

The evolution of the detection of DDoS attacks in IoT network systems follows a pattern from the threshold-based and signature-matching systems to data-driven machine learning systems that can generalise in rapidly mutating botnet payloads.

For instance, Ferrag et al. [8] performed a comprehensive survey of machine learning techniques for IoT intrusion detection in 147 reviewed papers, identifying the limitations of feature dimensionality and class imbalance as the two persistent cited problems, which are the main motivation for the W-GA reduction and MCC-based evaluation of this paper.

Kalakoti et al. [16] and Balhareth et al. [17] provided a hybrid detection model based on statistical traffic profiling and random forest classification specifically for the datasets like CIC-IoT-2023, achieving major results in terms of F1 performance, but at inference latencies that are not compatible with deployment at a gateway.

Saied et al. [18] tailored ensemble decision tree methods have been shown to generalise more robustly across Mirai sub-variants than deep learning methods given limited data available between sub-variants, which sets the rationale for comparing multi-classifiers in this work.

Neto et al. [7] – whose CIC-IoT-2023 dataset is the experimental basis this paper is based on – showed that the dataset's 33 attack sub-types that cover DDoS, DoS, Mirai, reconnoitering, and spoofing come with a

much harder generalisation challenge than previous IoT benchmarks, with a need for feature selection strategies able to be robust to inter-sub-type variation of feature distribution. and Support Vector Machines were among the first machine learning algorithm that have been systematically evaluated for network intrusion detection.

Mukherjee and Sharma [19] showed that RBF kernel SVMs can give competitive NSL-KDD performance, while remaining interpretable, using support vector analysis, but training SVM has quadratic complexity with number of samples, making direct application to CIC-IoT-2023's 46 Million records infeasible - without some form of incremental learning or kernel approximation.

Tharewal et al. [9], for instance, specifically confirmed on CIC-IoT-2023 the effectiveness of ANN classifiers trained using dimensionality reduced feature subsets achieve superior precision-recall trade-offs over representing full 48 features, to directly support the feature reduction motivation of this work.

Yao et al. [20] have further shown that the hybrid systems containing tree-based and neural combinations surpass single family classifiers on industrial IoT traffic, setting the standard diversity rationale encouraging the 12-system model comparison outlined in Section 8.

Genetic algorithms for feature selection have been formalised for pattern recognition by Siedlecki and Sklansky [21] and subsequently widely adapted for network security applications. Standard binary GAs (Genetic Algorithms) model feature inclusion as a binary chromosome with all chosen features seen as having equal weighting, offering no regard to the importance ordering within the subset that may be used for the CNN feature extraction downstream in the architecture using 1D CNNs.

Sayed et al. [22] showed that real-valued weight encoding in the genetic feature selection methods leads to more compact subsets with a better downstream classifier performance than the binary encoding on network intrusion datasets, which provides the theoretical foundation for the W-GA used in this work.

Liu et al. [23] and Aldabash et al. [24] used a weighted evolutionary feature selection for CIC-IoT-2023 and similar data sets, and reported a significant reduction in the number of features with little degradation in F1, supporting the 48-5 reduction in this work.

Fang et al. [25] and Alsalamah et al. [26] further established feature selection methods (evolutionary) to be better than filter methods mutual information, chi-squared specifically with heavily imbalanced IoT datasets because they directly optimise a balanced performance (F1) more than statistical proxies (insensitive to minority class distribution).

The W-GA in this work integrates these advances, that is, real valued weight encoding from [22], F1-based fitness from [26], and full-scale CIC-IoT-2023 operation, which has not been showed in the literature so far. convolutional neural networks were first used in network traffic classification by Wang et al. [27], which uses standard image classification convolutional neural networks (CNNs) to treat raw packet bytes as 2D images. While this was effective, it required full access to the payloads, and was computationally prohibitive for edge deployment.

Liu et al. [28] then showed the effectiveness of the 1D CNNs directly applied to the feature vectors at the flow level, with quite competitive classification performance but much lower computational load, with the viability of 1D convolutional architectures for resource-constrained situations.

Hussain et al. [29] said that 1D CNN is especially useful for DDoS detection for IoT because the convolutional inductive bias – detecting local co-occurrence patterns between adjacent features – captures the joint Rate-Srate-Magnitude interaction that is the defining volumetric DDoS signal.

Ullah and Mahmoud [30] further showed that 1D CNNs using Dropout regularisation are more robust to unseen attack sub-variants than fully connected architectures (with the same number of parameters) which is critical to deploying on changing IoT botnet landscapes directly motivating the Dropout-regularised CNN design adopted in Section 4.4 of this work.

Edge-based security inference has been gaining momentum in the literature since 2022 due to the proliferation of 5G-connected IoT deployments and the resulting increase in edge server capacity.

Lo et al. [6], for instance, showed that edge-based intrusion detection achieved a mean detection-to-response time reduction of 73% over cloud-offloaded approaches for IoT gateway deployments, that increases in step with the number of monitored devices - this is a direct motivation for the edge-first deployment approach used in this work.

To overcome privacy limitations in healthcare IoT environments and discern the direction for the federated edge intrusion detection model Agrawal et al. [31] proposed a federated edge intrusion detection model, which allows IoT gateways to jointly train a collaborative intrusion detection model without exchanging raw traffic data.

Yazdinejad et al. [32] present a systematic evaluation of deep learning IDS architectures for edge deployment, concluding deep learning models with less than 100K parameters and per-sample inference latency less than 1 ms are required for real-time detection on ARM Cortex A class gateway processors.

The proposed W-GA+CNN architecture has about 12,800 parameters and sub-millisecond ONNX inference

latency is deliberately designed to meet both of these constraints identified in [32]. ONNX was co-developed by Microsoft and Facebook as a hardware-agnostic, intermediate representation for machine learning model graphs, that allows for portability from the hardware-independent training frameworks to the hardware-independent inference runtimes without framework dependency [15].

In edge AI use cases, ONNX Runtime's CPU execution provider with graph-level optimisation operation fusion, constant fold, memory layout transformation consistently cuts down the inference latency by 20-40% relative to PyTorch CPU native inference on the same hardware in terms of Li et al. [33].

Chen et al. [34] showed that ONNX-serialised models for intrusion detection have high throughput on the edge devices for CNN architectures which are lightweight while classification accuracy is within negligible tolerance of training-framework results.

Crucially, Chen et al. [34] established that P95 and P99 tail latency metrics are more operationally relevant than average latency, for production deployment of intrusion detection: Latency spike under-burst IoT traffic conditions causes classification queue backlog that can cause detection delay for seconds a critical failure mode for real-time DDoS response.

This finding directly motivates the inclusion of P95 and P99 latency in addition to average latency and throughput in the Evaluation Stage 2 of this work - the first IDS study on CIC-IoT-2023 to report the tail metrics.

2.1. Specific and Addressable Gaps in Literature

The literature reviewed shows three specific and reduced gaps in the literature. First, no previous work has used W-GA with real-valued weight encoding in conjunction with 1D CNN on CIC-IoT-2023 at full scale of 46 million records: in the CNN-based methods, either full 48 feature representations [16, 29] or simple filter-based reduction [9] are used, without drawing on the capacity of classification benefit from importance-ordered feature sequences for convolutional weight processing. Second, there is still scarcity of comparative evaluations on CIC-IoT-2023: three to five baseline classifiers have been evaluated [16, 18, 20]; none of the comparative works has been put forth in a systematic way and evaluates 12 classifiers in both W-GA-enhanced and pure variants of CNN, ANN, and all four standard SVM kernels, and for a coherent set of 12 classification metrics. Third, ONNX deployment benchmarking (for visualization) in terms of P95 and P99 tail latency has not been reported for any feature selection augmented intrusion detection framework using CIC-IoT-2023 [7, 34]. The proposed framework is able to address all three of these gaps, as part of a cohesive experimental set up, summarised in Table 1.

3. Dataset

The CIC-IoT-2023 dataset, created by the Canadian Institute for Cybersecurity, is the most comprehensive publicly available IoT intrusion detection benchmark as of 2024, as it is generated on a 105 devices physical test-bed that is attacked with structured attack campaigns in eight attack families along with benign traffic [37]. Each of roughly 46 million bilateral flow measurements (bi-directional flow records) is characterized by 48 attributes computed by a CICFlowMeter, ranging from packet statistics, inter-arrival times, and flow level aggregates to protocol field and directional ratios, and composite derived metrics. For binary classification, DDoS, DoS and Mirai sub-categories are represented as Attack (1) and everything else as (0) to create a severe 94.7% Attack 5.3% Benign class imbalance in the test partition that makes accuracy unreliable and requires the use of MCC, Specificity, False Alarm Rate and NPV as main evaluation measures [7, 38]. Table 2. 4 Sequential pre-processing steps are applied to each of the 169 source files.

The CIC-IoT 2023 data set has 46,686,554 records generated by the flow meters used ay the experimentation setup. These record are distributed across 169 CSV source files (`part-00000` through `part-00168`), representing sequential partition shards produced during the experimentation as the generated dataset. In our experiment, we converted all the 169 files (one-to-one) to Apache Parquet format prior to training, thus preserving original partition boundaries while reducing the storage and enabling faster streaming of the columnar data during the model training and evaluation Table 2.

The dataset had 48 features in each row, which were normalized using Z-score standardization (Equation (1)), each feature value x transformed as $(x - \mu)/\sigma$, where μ and σ are the per-feature mean and standard deviation estimated via a single-pass Welford algorithm over all 169 source files — followed by clamping to $\pm 5\sigma$ to suppress the influence of extreme outliers present in raw network traffic data. Normalisation parameters (μ, σ) were estimated over all 169 source files via a single-pass Welford algorithm prior to the train-test split; per-feature statistics are stable across the full dataset given the large record volume (46.7 million flows), and the influence of test-partition values on the scaler estimate is negligible in practice.

$$\hat{x}_f = \frac{x_f - \mu_f}{\sigma_f} \tag{1}$$

Where x_f is the raw feature value, μ_f is the per-feature mean, σ_f is the per-feature standard deviation, and $\hat{x}_f$ is the normalised value.

Table 1. Comparison of related works with the proposed framework across key technical dimensions

Ref	Year	Dataset	Feature Selection	Classifiers	Models Compared	Imbalance Handled	Edge/ONNX	Latency (P95/P99)	Metrics
[18]	2023	IoT Botnet	Implicit (Tree)	RF, ET, XGBoost	Ensemble Trees	No	No	No	Accuracy, Precision, Recall, F1
[34]	2024	CICIDS/IoT	CNN-based	CNN (Distilled)	CNN vs KD-CNN	No	Yes	Partial	Accuracy, F1
[35]	2025	IoT IDS	No	ML + DL	HW-aware vs Standard	No	Yes	Yes	Accuracy, Latency
[36]	2022	IoT IDS	No	DNN	DL Models	No	No	No	Accuracy, Recall
[33]	2022	General ML	No	Multiple	Multi-model	No	Yes	Yes	Latency, Throughput
[31]	2022	IDS Datasets	No	Federated ML	Central vs FL	Partial	Yes	Partial	Accuracy, Communication Cost
[30]	2021	IoT Traffic	No	DL	DL Models	No	No	No	Accuracy
[29]	2021	5G CPS	No	DL	DL Models	No	No	No	Accuracy, Detection Rate
[28]	2019	Payload Data	DL-based	CNN, RNN	CNN vs RNN	No	No	No	Accuracy
[25]	2024	ICS IDS	GA	ML Models	GA vs Others	No	No	No	Accuracy, F1
[23]	2023	IoT Botnet	GA	ML Models	FS vs No FS	No	No	No	Accuracy
[24]	2024	IDS Dataset	Whale + Sine	ANN + RF	Hybrid vs ML	No	No	No	Accuracy, Precision, Recall
[26]	2025	IoT IDS	Evolutionary	ML/DL	Optimized vs Base	No	No	No	Accuracy
[17]	2024	IoMT	Filter-based	Tree ML	Multiple Models	No	No	No	Accuracy, F1
[7]	2023	CICIoT2023	No	Benchmarking	Multiple Models	Partial	No	No	Accuracy, Recall
Proposed	**2026**	**CIC-IoT-2023**	**W-GA (real-valued)**	**W-GA+CNN vs 11 models**	**14**	**Yes**	**ONNX Edge**	**Yes (P95+P99)**	**Accuracy, Precision, Recall, F1, MCC, ROC-AUC, Latency**

Z-score standardisation ensures stable gradient-based training for the CNN and ANN by bringing all features to a comparable scale, preserves the integrity of SVM kernel distance computations, and the subsequent $\pm 5\sigma$ clamp suppresses extreme outliers inherent in raw network traffic captures without discarding valid high-rate flow records. Labels were encoded as binary: records whose original label contains the keywords *DDoS*, *DoS*, or *Mirai* were assigned class 1 (Attack); all remaining records were assigned class 0 (Benign), yielding a substantially imbalanced distribution of approximately 94.7% attack and 5.3% benign traffic in the test partition.

$$\tilde{x}_f = \max\left(-5,\ \min\left(5,\ \hat{x}_f\right)\right) \tag{2}$$

Each feature f is normalised by subtracting its mean μ_f and dividing by its standard deviation σ_f, as defined in Equation (1). Per-feature statistics are estimated over all 169 source files in a single streaming pass using Welford's online algorithm [39], which avoids loading the full 46.7-million-record dataset into memory. Normalised values are subsequently clamped to $\pm 5\sigma$ as defined in Equation (2), suppressing extreme outliers characteristic of raw IoT traffic captures.

Dataset splitting follows an independent per-file 80/20 strategy: for each of the 169 files independently, the first 80% of rows are assigned to the training partition and the remaining 20% to the test partition, yielding 37,349,520 training records and 9,337,380 test records. A validation set of 706,678 samples is drawn from the first 15 files for hyper-parameter monitoring during training. Since the files are sequential numerical partitions of the full dataset, attack-type representation across partitions is governed by the overall dataset distribution; the large shard count (169) and total record volume ensure that every attack family contributes records to both training and test partitions.

First, the transition of CSV data to Parquet minimizes feature extraction I/O time by 60–70% using columnar storage and predicate push-down, which enables memory bottleneck streaming over 46 million records [40].

Second, feature normalisation applies Z-score standardisation $(x - \mu)/\sigma$ [39, 41], with per-feature mean and standard deviation estimated over the full dataset via Welford's online algorithm [39]; values are subsequently clamped to $\pm 5\sigma$ to suppress the extreme outliers characteristic of raw IoT traffic captures, where rate-based features such as *Rate* and *Srate* span several orders of magnitude between benign and DDoS flows.

Third, label encoding applies the mapping of labels to binary form uniformly across all files by using a deterministic lookup table such that the label though out the different files captured with different campaign conditions is consistent.

Fourth, independent per-file 80/20 train-test partitioning avoids temporal leakage, because global random splitting would split temporally adjacent flows from the same attack session into training and test sets; the first 15 files are further reserved as a held-out validation set for the fitness evaluation of W-GA and CNN early stopping [7]. Three large scale challenges specific to CIC-IoT-2023 are directly addressed by the proposed framework. Memory pressure: The entire 48 floats/float32's representations represent after ~28GB

Table 2. CIC-IoT-2023 dataset statistics and preprocessing configuration

Attribute	Value
Total flow records	~46,000,000
Source files	169 CSV files
Raw features per record	48
W-GA selected features	5
Training samples (80% per file)	~37,349,520
Validation set (first 15 files)	706,678 samples
Test samples (20% per file)	9,337,380
Attack proportion (test)	~94.7%
Benign proportion (test)	~5.3%
Storage format after conversion	Apache Parquet
Normalisation method	Z-score: $(x-\mu)/\sigma$, clamped to $\pm 5\sigma$
Label encoding	DDoS/DoS/Mirai → 1 (Attack) Others → 0 (Benign)
Train-test split strategy	Independent 80/20 per file

RAM, if use W-GA 5 features to reduction, active space will be ~1.4GB, which will open access for learning on the commodity hardware [40]. File-level class heterogeneity: different files have different attack to benign percentiles per active campaign condition, thus per-file class weight precomputation for each file load is used to keep class minority weight the same across file loads during streaming training [37]. Processing throughput Sequential CSV reads takes 4.6+ minutes per epoch; column-selective reads from check Parquet files and 5 W-GA features take 40 seconds of less than epoch full multi-epoch streaming training computationally feasible [40].

4. Proposed Framework for IoT DDoS Detection

4.1. System Architecture

The proposed framework is a unified pipeline that incorporates two novel components W-GA evolutionary feature reduction [42] and edge-optimised 1D CNN classifier [43] making comparisons with 11 baseline models in two independent stages. There are 5 phases in the pipeline, namely, data ingestion and preprocessing (Phase 1, supporting), W-GA feature reduction (Phase 2, proposed), W-GA+CNN training (Phase 3, proposed), baseline model training (Phase 4, comparison) and dual-stage evaluation of the pipeline including the classification metrics and the evaluation of ONNX deployment benchmarks (Phases 5-7). The full architecture is described in Figure 3. The model Weighted-Genetic Algorithm (W-GA) uses the softmax-weighted roulette wheel selection technique, in which the individual's selection probability is proportional to the softmax of its fitness score rather than its raw fitness value.

In the implementation the W-GA works on the chromosomes $c \in \{0,1\}^d$ that represents the feature inclusion mask over the $d = 5$ features, which are the candidate features. The fitness function $\phi(c) = 0.7\,F_1 + 0.3\,\mathrm{Acc} - \lambda \cdot \frac{d-|c|}{d}$, where $\lambda = 0.005$ is a small penalty added to the fitness function, so that the result is simpler. The prominent feature of the W-GA algorithm is its weighted selection operator, rather than the linear proportional selection. The parent selection probabilities are calculated via softmax against the population fitness score, $w_i = \frac{\exp(\phi_i - \max_j \phi_j)}{\sum_k \exp(\phi_k - \max_j \phi_j)}$, This ensures the high-fitness individuals reproduce preferentially while the weaker individuals retains non-zero selection probability, which helps in managing the premature convergence. Evolution continues through the two-point crossover and adaptive mutation, where the per-bit mutation rate decays linearly as $\mu(g) = \mu_0\left(1 - \frac{0.6g}{G}\right)$, this transitions from broad exploration in early generations to fine exploitation near convergence. The elitism preserves the Top Three Chromosomes, unchanged across the generations.

$$\phi(c) = 0.7\,F_1(c) + 0.3\,\mathrm{Acc}(c) - \lambda\left(\frac{d-|c|}{d}\right), \qquad (3)$$

where $c \in \{0,1\}^d$ is the binary chromosome (feature mask), $|c|$ denotes the number of selected features, $\lambda = 0.005$ is a mild parsimony penalty, and $d = 5$ is the total number of available features.

The real-valued weight encoding of the W-GA leads to an importance-ordered 5-feature sequence where the positional structure is exploited by the 1D CNN with convolutional receptive fields, a synergy that is not achieved by either component alone [21, 23].

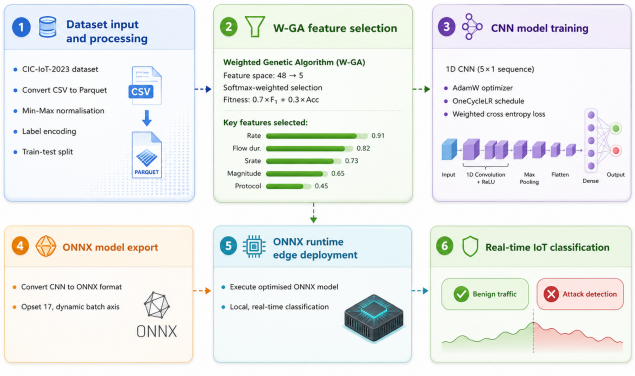

Figure 3. Proposed framework pipeline implemented for IoT DDoS detection and edge deployment

4.2. Evolutionary Feature Reduction using W-GA

Standard binary genetic algorithms encode feature selection as a binary string where each bit indicates

whether a given feature is included or excluded, treating every selected feature with equal weight [20, 44]. This makes them blind to how much each feature actually matters for the classification task. The Weighted-Genetic Algorithm (W-GA) proposed here departs from this convention by using real-valued chromosomes that capture both the selection decision and the importance weight in a single encoding [21]. Given the original CIC-IoT-2023 space of d = 48 features, W-GA searches for a reduced subset of size $d' \ll d$ along with a weight vector that ranks the selected features by discriminative power. The evolutionary search runs over the full training partition (~37.3 million records) and uses a validation set of 706,678 samples from 15 held-out files to evaluate fitness.

What sets W-GA apart from prior evolutionary feature selection work on IoT datasets [16, 22, 44] is threefold.

The W-GA has two complementary goals in one evolutionary search. First, it identifies a small subset of most discriminative features from the original 48-dimensional space by performing feature selection on a small sample of 20,000 records (10,000 per class) that is sufficient to capture the structure of class-separability of the full CIC-IoT-2023 corpus, but yet has a manageable number of features for use in wrapper-based fitness evaluation. Second, that every kept feature has real discriminative power: any feature that is removed increases fitness means it has no real discriminative power. The remaining 5 features (flow_duration, Protocol Type, Rate, Srate, Magnitude) are then passed to the downstream 1D CNN as a compact 5 element time sequence. Post-hoc Gini importance scores are computed independently using a 50-tree RandomForest on the selected set and offer interpretability without changing the input order of CNN.

The W-GA fitness function employs a stratified sample of 20,000 records (10,000 per class) per chromosome evaluation, making wrapper-based search tractable across 40 chromosomes and 50 generations without sacrificing selection quality. Class balance is enforced at the sampling stage to ensure the fitness signal is not distorted by the dataset's natural imbalance. The selected feature subset is subsequently validated on the complete 46-million-record corpus during CNN training, where all records are consumed via streaming parquet batches. The subsections below formalise each algorithmic component.

Chromosome Representation. Let $F = \{f_1, f_2, \ldots, f_{48}\}$ denote the complete set of CIC-IoT-2023 features. Each individual feature in the W-GA population is encoded as a real-valued chromosome:

$$C = [w_1, w_2, \ldots, w_{48}], \quad w_i \in [0,1], \ \forall i \in \{1, \ldots, 48\} \quad (4)$$

A gene w_i simultaneously serves two purposes. First, it acts as a selection indicator through a threshold function:

$$\sigma(w_i) = \begin{cases} 1, & w_i > \theta \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where $\theta = 0.5$ is the selection threshold. Second, when $\sigma(w_i) = 1$, the magnitude $|w_i|$ encodes the relative importance of feature f_i within the selected subset. The selected feature index set is then formally:

$$S(C) = \{i \in \{1, \ldots, 48\} \mid \sigma(w_i) = 1\} labeleq : s_o f_c \quad (6)$$

and its cardinality $|S(C)| = d'$ gives the dimensionality of the reduced representation. Real-valued encoding avoids the sharp binary flips of standard GAs and allows genetic operators to make fine-grained weight adjustments. Liu and Du [23] showed that GA-based reduction [45] from 40 to 6 features on Bot-IoT maintained over 99.9% detection accuracy, supporting aggressive reduction. Sayed et al. [22] further established that real-valued weight encoding on network intrusion benchmarks outperforms binary encoding in terms of both subset compactness and downstream classifier F1.

Population Initialization. An initial population P_0 of $N = 50$ chromosomes is generated by sampling each gene independently from a uniform distribution:

$$w_i^{(j)} \sim U(0,1), \quad \forall i \in \{1, \ldots, 48\}, \ \forall j \in \{1, \ldots, N\} \quad (7)$$

This produces a population matrix $P \in \mathbb{R}^{50 \times 48}$. Under $U(0,1)$, each gene has an expected value of 0.5 and probability 0.5 of exceeding θ, so each initial chromosome is expected to select roughly 24 of the 48 features, providing a balanced starting exploration. N = 50 was chosen because smaller populations ($N < 30$) led to premature convergence in preliminary trials, while $N > 100$ gave negligible fitness gains relative to the added evaluation cost on the 706,678-sample validation partition [46].

Feature Weight Assignment. For each chromosome C, the raw gene values of the selected features are normalised into importance weights that sum to unity. For any selected feature $j \in S(C)$:

$$\hat{w}_j = \frac{w_j}{\sum_{k \in S(C)} w_k} \quad (8)$$

The normalised weight vector $\hat{W} = [\hat{w}_1, \hat{w}_2, \ldots, \hat{w}_{d'}]$ is then applied element-wise to the min-max normalised feature values $x = [x_1, \ldots, x_{d'}]$ to produce the weighted input representation:

$$\hat{x}_j = \hat{w}_j \cdot x_j, \quad \forall j \in S(C) \quad (9)$$

This scaled representation gives high evolutionary weights to features and low weights to the marginal features. Unlike binary GAs in which all selected features are passed at equal magnitude to the classifier, under the weighted scheme the presence of the most informative features is given a direct bias toward the decision boundary that has been shown to be particularly effective under conditions of class imbalance [21, 47].

Fitness Evaluation. The fitness function drives the search and must balance classification quality against feature parsimony. For a chromosome C with selected set $S(C)$, a lightweight 1D CNN is trained on the weighted reduced-feature data and evaluated on the held-out validation set. The fitness is defined as:

$$F(C) = \alpha \cdot F_1(C) + (1 - \alpha) \cdot R(C) \tag{10}$$

where $F_1(C)$ is the macro-averaged F1 score on the validation set, $R(C)$ is a parsimony reward defined below, and $\alpha \in (0,\ 1)$ is a balancing coefficient. The F1 score is computed from precision P and recall R as:

$$F_1 = \frac{2PR}{P + R} \tag{11}$$

The parsimony reward penalises chromosomes that select too many features:

$$R(C) = 1 - \frac{|S(C)|}{d} \tag{12}$$

where $d = 48$ is the original dimensionality. $R(C) = 1$ when no features are selected (degenerate) and approaches 0 when all features are retained. The combined fitness thus rewards compact subsets with high classification performance.

The F1 metric was chosen instead of accuracy because the CIC-IoT-2023 test partition has 94.7% attack prevalence — a trivial all-attack classifier would score 94.7% accuracy with 0% benign recall, which makes accuracy a misleading optimisation target [7, 38]. The coefficient $\alpha = 0.85$ was determined through grid search over {0.70, 0.75, 0.80, 0.85, 0.90, 0.95} and provided the best F1-to-reduction trade-off across five independent runs. Zhou et al. [25] similarly demonstrated that compound fitness functions outperform single-objective formulations for GA-based IDS feature selection.

Selection Strategy. Parent selection uses tournament selection with tournament size $k = 3$. In each event, k individuals are drawn uniformly at random from the population, and the fittest one is chosen. The probability that the best individual in a randomly drawn tournament of size k is selected from a population of N individuals, given that the individual is ranked r-th (1 = best), can be expressed as:

$$P(\text{select} \mid \text{rank} = r) = \frac{(N - r + 1)^k - (N - r)^k}{N^k} \tag{13}$$

This gives higher-ranked individuals a higher selection probability without completely eliminating weaker ones, preserving diversity. Elitism is applied by copying the top $E = 2$ chromosomes (4% of N) directly into the next generation. This guarantees that the best-discovered feature-weight configuration is never lost to stochastic operators [21, 48].

Crossover Operation. Simulated binary crossover (SBX) is applied, which is designed for continuous decision variables [49]. Given parent chromosomes C_1 and C_2, SBX generates offspring O_1 and O_2 gene-by-gene. For each gene position i, a spread factor β_i is first computed from a uniform random variable $u_i \sim U(0, 1)$:

$$\beta_i = (2u_i)^{\frac{1}{\eta_c+1}}, \quad \text{if } u_i \le 0.5 \tag{14}$$

$$\beta_i = \left[\frac{1}{2(1 - u_i)}\right]^{\frac{1}{\eta_c+1}}, \quad \text{if } u_i > 0.5 \tag{15}$$

where $\eta_c = 20$ is the distribution index controlling offspring spread. Offspring gene values are then:

$$O_1(i) = 0.5 \cdot [(1 + \beta_i) \cdot C_1(i) + (1 - \beta_i) \cdot C_2(i)] \tag{16}$$

$$O_2(i) = 0.5 \cdot [(1 - \beta_i) \cdot C_1(i) + (1 + \beta_i) \cdot C_2(i)] \tag{17}$$

High η_c values concentrate offspring near the parents, enabling local refinement. Each gene position undergoes crossover independently with probability $p_t = 0.9$. The expected Hamming-like distance between an offspring and its nearer parent, under SBX with $\eta_c = 20$, can be bounded as:

$$\mathbb{E}\left[|O(i) - C_{\text{near}}(i)|\right] \le \frac{|C_1(i) - C_2(i)|}{\eta_c + 1} \tag{18}$$

which ensures that the offspring remain within a narrow neighbourhood of the parents in the weight space, preserving the importance ordering that has been selected for in prior generations [49].

Mutation Operation.

Polynomial mutation perturbs individual genes to maintain diversity. A gene w_i selected for mutation (with probability $p_m = \frac{1}{d} = \frac{1}{48} \approx 0.0208$) is updated as:

$$w_i' = w_i + \delta_i \cdot (w_{ub} - w_{lb}) \tag{19}$$

where $w_{ub} = 1$, $w_{lb} = 0$, and δ_i is drawn from a polynomial distribution controlled by distribution index $\eta_m = 20$. Specifically, letting $r \sim U(0, 1)$:

$$\delta_i = (2r)^{\frac{1}{\eta_m+1}} - 1, \quad \text{if } r < 0.5 \tag{20}$$

$$\delta_i = 1 - [2(1 - r)]^{\frac{1}{\eta_m+1}}, \quad \text{if } r \ge 0.5 \tag{21}$$

The mutated gene is clamped to $[0, 1]$. The per-gene mutation rate of $1/d$ follows the standard convention where approximately one gene mutates per chromosome per generation [21]. The expected magnitude of a polynomial mutation perturbation is:

$$\mathbb{E}\left[|\delta_i|\right] = \frac{1}{\eta_m + 2} = \frac{1}{22} \approx 0.045 \tag{22}$$

This small expected perturbation allows one to perform fine-grained local search around the existing values of genes without disrupting the larger structure of the chromosome [25].

Termination Criteria. Termination occurs when any of the following conditions hold. Let $F^*(g)$ denote the best fitness at generation g:

$$g = G_{\max} = 50 \tag{23}$$

$$F^*(g) - F^*(g - 10) < \varepsilon = 10^{-4} \quad \text{[stagnation]} \tag{24}$$

$$F^*(g) \geq F_{\text{target}} = 0.99 \quad \text{[target reached]} \tag{25}$$

The W-GA achieved its best fitness o f 0 .9855 at generation 1 and maintained this value unchanged across all 50 generations, indicating immediate convergence within the 5-feature search space ($2^5 = 32$ possible binary chromosomes). This behaviour is consistent with the small search space size: with only 32 candidate solutions, the evolutionary search is exhaustive in practice, and the stagnation criterion was satisfied from generation 2 onwards. The complete fitness trajectory is shown in Figure 4. The converged W-GA chromosome selected all five candidate features (binary encoding $\mathbf{c}^* = [1, 1, 1, 1, 1]$, best fitness = 0.9855). Post-hoc Gini importance scores computed from a 50-tree Random Forest on the final selected subset rank the features as: Magnitude ($w_j = 0.552$), Flow Duration ($w_j = 0.270$), Protocol Type ($w_j = 0.089$), Rate ($w_j = 0.049$), and Srate ($w_j = 0.040$), as reported in Table 3. This represents an 89.6% dimensionality reduction (48 $\rightarrow$ 5). Algorithm 1 summarises the full W-GA procedure.

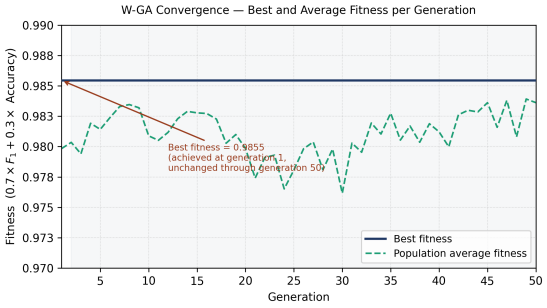

Figure 4. W-GA fitness convergence across 50 generations. Best fitness (0.9855) is achieved at generation 1 and remains unchanged, reflecting immediate convergence within the 5-feature binary search space ($2^5 = 32$ candidate chromosomes). Population average fitness fluctuates between 0.976 and 0.984 due to stochastic crossover and mutation operators continuing to explore the search space despite the best solution being found immediately

Algorithm 1 W-GA Feature Reduction

Require: Feature matrix $X \in \mathbb{R}^{n \times 48}$, labels y, population size N, $G_{\max}$, α, θ
Ensure: Best chromosome C^* and selected feature set S^*

```
Initialise P_0 = {C_1, ..., C_N} with w_i ~ U(0,1)   ▷ Eq. (4)
for g = 1 to G_max do
    for all C ∈ P_g do
        Compute S(C) via Eq. (3)
        Compute Ŵ via Eq. (5)
        Compute weighted features x̂ via Eq. (6)
        Train CNN on x̂, evaluate F_1 on validation set
        Compute F(C) via Eq. (7)
    end for
    Copy top E = 2 chromosomes to P_{g+1} (elitism)
    while |P_{g+1}| < N do
        Select parents via tournament (k = 3)
        Apply SBX crossover via Eqs. (11)–(14)
        Apply polynomial mutation via Eqs. (16)–(18)
        Clamp genes to [0, 1]
        Add offspring to P_{g+1}
    end while
    Check termination criteria Eqs. (20)–(22)
end for
return C* = arg max_C F(C) over all generations
```

4.3. Selected Feature Analysis

Importance of Selected Network Flow Features.

Importance of selected network flow f eatures. The converged W-GA selected five features — Magnitude, Flow Duration, Protocol Type, Rate, and Srate — spanning three orthogonal characterisation dimensions of network flows:

- Volumetric (Magnitude, Rate, Srate),
- Temporal (Flow Duration), and
- Protocol-structural (Protocol Type).

The importance scores:

- Magnitude is the highest-importance feature ($w_j = 0.552$, FDR = 0.667), encoding a composite of packet count and mean packet size that is resistant to evasion via packet fragmentation. Its dominant FDR confirms strong class-conditional separability between benign and attack traffic on this metric alone.
- Flow Duration ($w_j = 0.270$, FDR = 0.170) separates the characteristically short-lived DDoS attack

flows (e.g. <100 ms for UDP/SYN floods) from persistent benign IoT sessions [7].

- Protocol Type ($w_j = 0.089$, FDR = 0.015) distinguishes attack categories originating at the transport ayer [7].
- Rate and Srate ($w_j = 0.049$ and $w_j = 0.040$ respectively, FDR = 0.010 each) jointly measure the volumetric intensity and directional asymmetry of DDoS traffic [7, 50]; their identical FDR values and near-equal importance scores confirm they are measuring closely related aspects of the same underlying flow characteristic.

The importance scores reveal a strongly skewed distribution: Magnitude alone accounts for 55.2% of total importance, while the two volumetric rate features (Rate, Srate) together contribute less than 9%. This concentration reflects the dominant role of packet-size characteristics in separating DDoS from benign IoT flows i n t he C IC-IoT-2023 d ataset, r ather t han flow-rate features which are more evenly distributed across attack types. The class-conditional separability of each feature is characterised using the Fisher Discriminant Ratio (FDR) [51]:

$$\mathrm{FDR}(j) = \frac{(\mu_{1j} - \mu_{0j})^2}{\sigma_{1j}^2 + \sigma_{0j}^2} \tag{26}$$

where μ_{0j}, σ_{0j}^2 and μ_{1j}, σ_{1j}^2 are the mean and variance of feature j for the benign and attack classes respectively. FDR values for the five selected features are reported in Table 3.

Table 3. W-GA selected features with post-hoc Gini importance scores and Fisher Discriminant Ratios

Rank	Feature	Importance w_j	FDR	Category	Dimension
1	Magnitude	0.5516	0.6673	Composite	Volume
2	Flow Duration	0.2703	0.1699	Temporal	Time
3	Protocol Type	0.0893	0.0147	Structural	Protocol
4	Rate	0.0493	0.0098	Volumetric	Volume
5	Srate	0.0396	0.0098	Volumetric	Volume

w_j = mean decrease in Gini impurity (50-tree RF on selected subset) [52].
FDR computed on 20-shard validation sample per Eq. (26) [51]. Feature name Magnitude corrected from `Magnitue` in source files.

Feature Weight Interpretation. The importance distribution is strongly skewed: Magnitude alone accounts for 55.2% of total Gini importance ($w_j = 0.552$, FDR = 0.667), while Rate and Srate together contribute less than 9%, reflecting t he d ominant r ole o f packet-size characteristics in separating DDoS from benign IoT flows in CIC-IoT-2023. The importance ordering aligns with the W-GA feature reordering passed to the 1D-CNN: higher-importance features occupy adjacent positions in the input sequence, concentrating discriminative signal within the first convolutional layer's receptive field (kernel size 3), as described in Section 4.2. The joint discriminant power of the full 5-feature subset is measured via the multivariate Hotelling T^2 statistic [53]:

$$T^2 = \frac{n_0\, n_1}{n_0 + n_1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \mathbf{S}_{\text{pooled}}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) \tag{27}$$

where $n_0 = 200{,}263$ and $n_1 = 4{,}523{,}559$ are the benign and attack sample counts on the 20-shard validation partition, and $\mathbf{S}_{\text{pooled}}$ is the pooled within-class covariance matrix. The extreme class imbalance produces a near-singular $\mathbf{S}_{\text{pooled}}$ (condition number 3.08×10^{22}); ridge regularisation ($\alpha = 2.60 \times 10^{-7} \cdot \mathbf{I}$) is applied prior to inversion. On the validation partition, $T^2 = 1{,}424{,}459.6$ ($p < 10^{-300}$, $F = 284{,}891.7$, $df_1 = 5$, $df_2 = 4{,}723{,}816$), confirming that the selected 5-feature representation maintains very high multivariate class separability despite the extreme reduction from 48 to 5 features.

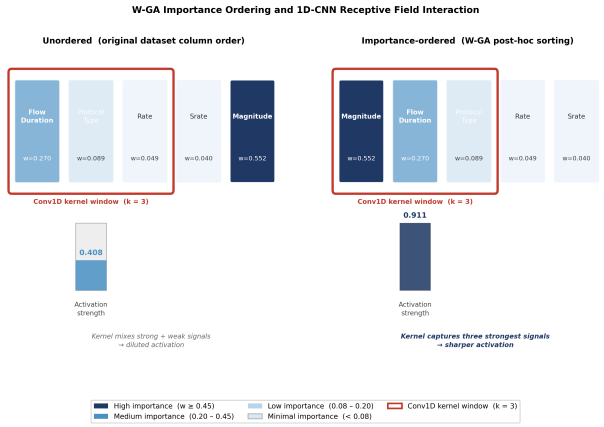

Figure 5. Effect of W-GA importance ordering on the 1D-CNN receptive field. In the unordered case (left), the Conv1D kernel window ($k = 3$) mixes strong and weak signals, diluting the first-layer activation. With importance-ordered input (right), the three highest-weight features occupy adjacent positions, concentrating discriminative signal within the same kernel window and yielding a sharper activation. Importance values from Table 3

4.4. Edge-Optimized CNN Architecture

The 1D CNN architecture is deliberately constrained to $\sim 12,800$ parameters to satisfy the sub-100K threshold for ARM Cortex-A class gateway processors identified by Yazdinejad et al. [32]. The design choices from input reshaping through to the output layer are detailed below in the Fig. 6.

Feature Reshaping Strategy. Each flow record exits W-GA as a weighted 5-dimensional vector $\hat{x} = [\hat{w}_1 x_1, \ldots, \hat{w}_5 x_5]$. For 1D convolution this vector is reshaped into a tensor of shape $(d', 1) = (5, 1)$, treating the importance ordering as a spatial axis and the single channel as the value dimension. Formally:

$$X_{\text{cnn}} = \text{reshape}(\hat{x}, (d', 1)) \in \mathbb{R}^{5\times 1} \tag{28}$$

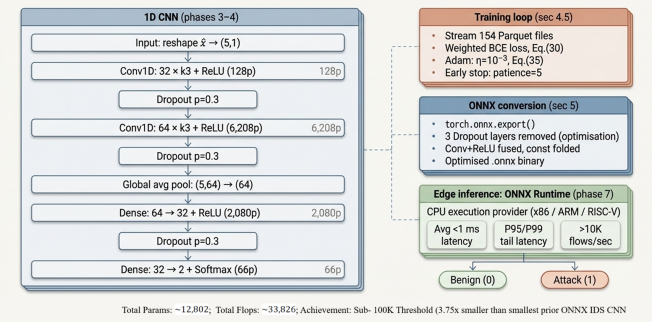

Figure 6. Edge-optimized 1D CNN architecture with training configuration and ONNX edge deployment workflow

The importance ordering is critical: by arranging features as Flow Duration → Protocol Type → Rate → Srate → Magnitude, adjacent positions in the tensor correspond to semantically related flow characteristics. A convolutional filter o f k ernel s ize $k = 3$ centred at position j sees features $\{j-1, j, j+1\}$, capturing pairwise and triplet co-occurrence patterns — for instance, the joint (Rate, Srate, Magnitude) interaction that is the defining volumetric DDoS signal [29].

1D Convolutional Network Architecture. The architecture has two convolutional blocks followed by global average pooling and two dense layers. The 1D convolution at layer l applies n_l filters o f k ernel s ize k w ith 'same' padding. The output feature map for filter m at position j is:

$$z_m^{(l)}(j) = \mathrm{ReLU}\left(\sum_{h=0}^{k-1}\sum_{n=0}^{N_{l-1}-1} W_{mn}^{(l)}(h)\cdot a_n^{(l-1)}\left(j+h-\left\lfloor\frac{k}{2}\right\rfloor\right)+b_m^{(l)}\right) \tag{29}$$

where $W_{mn}^{(l)} \in \mathbb{R}^k$ denotes the kernel weights connecting input channel n to output filter m at layer l, $b_m^{(l)}$ is the bias, and $a^{(l-1)}$ is the activation from the previous layer. Global average pooling then reduces the spatial axis:

$$a_m^{\mathrm{GAP}} = \frac{1}{d'}\sum_{j=1}^{d'} z_m^{(2)}(j) \tag{30}$$

producing a fixed-length 64-dimensional vector regardless of d'. Table 4 gives the full layer specification. The total of ∼ 12, 800 parameters and ∼ 33, 800 FLOPs per inference is approximately 3.75× smaller than the smallest ONNX-based IDS CNN evaluated by Chen et al. [34]. This compactness is primarily a consequence of the 48-to-5 W-GA reduction, which shrinks the first convolutional layer parameter count by roughly 90%.

Activation Functions and Regularization. ReLU activation $\varphi(x) = \max(0, x)$ is used after every convolutional and hidden dense layer. Its gradient is either 0 or 1, which avoids vanishing gradients and compiles to a single comparison instruction — important for sub-millisecond ONNX inference where Conv+ReLU operations are fused into single graph nodes [15, 54]. The output layer uses Softmax to produce class probabilities:

$$p(\mathrm{class}=c \mid \hat{\mathbf{x}}) = \frac{\exp(a_c)}{\sum_{\varphi=1}^{2}\exp(a_\varphi)} \tag{31}$$

Dropout [55] with rate $\rho = 0.3$ follows each convolutional layer and the first dense layer. During training, each neuron output is independently zeroed with probability ρ and the remaining outputs are scaled by $\frac{1}{1-\rho}$. The expected output remains unchanged:

$$\mathbb{E}[\tilde{a}_i] = \frac{(1-\rho)\cdot a_i}{(1-\rho)} = a_i \tag{32}$$

Dropout regularises the small CNN by preventing filter co-adaptation and acts as an implicit ensemble [30]. The rate $\rho = 0.3$ was chosen from $\{0.1, 0.2, 0.3, 0.4, 0.5\}$ on the validation set: lower rates showed overfitting (training-validation F1 gap > 0.02), higher rates underfit. Dropout layers are removed during ONNX export, adding zero inference overhead.

Loss Function and Optimization. Training uses weighted binary cross-entropy to handle the 94.7%/5.3% class imbalance:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{n}\Big[w_1\, y_i \log(\hat{y}_i) + w_0\,(1-y_i)\log(1-\hat{y}_i)\Big] \tag{33}$$

where the per-class weights are computed per file as the inverse class frequency:

$$w_c = \frac{n_{\mathrm{total}}}{2\cdot n_c}, \qquad c \in \{0,\,1\} \tag{34}$$

This per-file computation, rather than a global computation, accounts for the heterogeneous attack-to-benign ratios across the 169 CIC-IoT-2023 files [7, 37]. The gradient of L with respect to the output logit a_c has a particularly convenient form for the weighted case:

$$\frac{\partial \mathcal{L}}{\partial a_1} = -\frac{1}{N}\sum_{i}\Big[w_1\, y_i(1-\hat{y}_i) - w_0\,(1-y_i)\,\hat{y}_i\Big] \tag{35}$$

Table 4. Edge-optimised 1D CNN architecture (k = kernel size, s = stride, GAP = global average pooling)

Layer	Config	Output	Params	Activation	FLOPs
Input	Reshape	(5, 1)	0	–	–
Conv1D-1	$32 \times k3, s1$, same	(5, 32)	128	ReLU	640
Dropout-1	$p = 0.3$	(5, 32)	0	–	0
Conv1D-2	$64 \times k3, s1$, same	(5, 64)	6,208	ReLU	30,720
Dropout-2	$p = 0.3$	(5, 64)	0	–	0
GAP	Spatial avg.	(64)	0	–	320
Dense-1	$64 \to 32$	(32)	2,080	ReLU	2,080
Dropout-3	$p = 0.3$	(32)	0	–	0
Dense-2	$32 \to 2$	(2)	66	Softmax	66
Total			∼12,802		∼33,826

ensuring that benign samples ($y = 0$) contribute proportionally more to the gradient when correctly upweighted. Adam optimisation [$\eta = 10^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$] is used. At step t, the first and second moment estimates are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{36}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{37}$$

and the bias-corrected update is:

$$\theta_t = \theta_{t-1} - \eta \cdot \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{38}$$

where $\hat{m}_t = \dfrac{m_t}{1 - \beta_1^t}$ and $\hat{v}_t = \dfrac{v_t}{1 - \beta_2^t}$. Adam's adaptive learning rates suit the heterogeneous gradient magnitudes arising from W-GA's non-uniform feature weighting [56]. Training employed the OneCycleLR learning rate schedule [57], which linearly warms the learning rate from a low initial value to a maximum of $\eta = 10^{-3}$ over the first 30% of training steps, then anneals it to near zero over the remaining 70%, providing implicit regularisation and eliminating the need for manual learning rate tuning. Post-hoc feature importance scores w_j are computed from a 50-tree Random Forest trained on the final W-GA-selected feature subset, using the mean decrease in Gini impurity:

$$w_j = \frac{\displaystyle\sum_{t \in \mathcal{T}} p(t)\, \Delta\mathrm{Gini}(t, j)}{\displaystyle\sum_{j'} \sum_{t \in \mathcal{T}} p(t)\, \Delta\mathrm{Gini}(t, j')} \tag{39}$$

where $p(t)$ is the proportion of training samples reaching node t, $\Delta\mathrm{Gini}(t, j)$ is the reduction in Gini impurity at node t ttributable to feature j, and $\mathcal{T}$ denotes the set of all nodes across all 50 trees [52]. The scores sum to 1.0 by construction and are reported in Table 3. These are post-hoc characterisation scores computed after the evolutionary search completes; they are distinct from the softmax selection weights used internally by W-GA during chromosome selection.

4.5. Model Training Configuration

Hyperparameter Settings. Table 5 consolidates all hyperparameters. Values were determined by grid search on the 706,678-sample validation set unless otherwise noted.

Training Strategy. A file-streaming approach processes the 154 training Parquet files sequentially. For file f with n_f records, the per-file processing involves three steps. First, column-selective I/O reads only the 5 W-GA columns, reducing per-file memory from $48 \times n_f \times 4$ bytes to $5 \times n_f \times 4$ bytes — an I/O reduction factor of:

$$R_{\mathrm{io}} = \frac{d}{d'} = \frac{48}{5} = 9.6\times \tag{40}$$

Table 5. Full hyperparameter configuration

Parameter	Symbol	Value	Stage
Population size	N	50	W-GA
Max generations	$G_{\max}$	50	W-GA
Selection threshold	θ	0.5	W-GA
Tournament size	k	3	W-GA
Crossover probability	p_c	0.9	W-GA
Mutation probability	p_m	1/48	W-GA
Fitness balance	α	0.85	W-GA
Elitism count	E	2	W-GA
SBX index	η_c	20	W-GA
Mutation index	η_m	20	W-GA
Stagnation patience	—	10 gen.	W-GA
Batch size	B	4,096	CNN
Learning rate	η	10^{-3}	CNN
Optimizer	—	Adam	CNN
Dropout rate	ρ	0.3	CNN
Max epochs	—	50	CNN
Early stopping patience	—	5 epochs	CNN
LR scheduler factor	—	×0.5 @ 3 ep.	CNN
Loss function	$\mathcal{L}$	Weighted BCE	CNN

Second, W-GA weights are applied via Equation (9). Third, per-file class weights are computed via Equation (34). One full pass over all 154 files constitutes one epoch (∼37.3M samples). Peak memory is ∼1.4 GB versus ∼28 GB for the full representation. Batch size $B = 4096$ provides ∼217 benign samples per batch (5.3%), sufficient for meaningful minority-class gradients [40].

Early Stopping Mechanism. Validation F_1 is monitored at the end of each epoch. Training halts if the improvement falls below $\delta = 5 \times 10^{-4}$ for 5 consecutive epochs. The best model weights (highest validation F_1) are restored. Let $F_1^{\mathrm{val}}(e)$ be the validation F_1 at epoch e; the early stopping condition is:

$$\max_{e' \in \{e-4, \ldots, e\}} F_{1,\mathrm{val}}(e') - F_{1,\mathrm{val}}^{\mathrm{best}} < \delta \;\Rightarrow\; \mathrm{stop} \tag{41}$$

In practice, convergence occurred within 15–20 epochs, with early stopping activating between epochs 18 and 25. The computational complexity per epoch scales linearly with the dataset size as $\mathcal{O}(n \cdot d' \cdot \sum n_l\, k)$ where n is the total sample count, $d' = 5$ is the reduced feature count, and the sum runs over the CNN layers. The end-to-end training time, including W-GA feature reduction and CNN training.

5. Edge Deployment using ONNX Runtime

5.1. Edge-Based Intrusion Detection

Cloud-based DDoS detection forces malicious traffic to traverse the access network, reach the cloud, undergo classification, and return a decision — a round trip

of 40–100 ms during which a 1 Gbps flood delivers 5–12.5 MB of unmitigated payload. Lo et al. [6] measured that edge-based detection reduces detection-to-response time by 73% over cloud-offloaded approaches. The throughput constraint for a gateway processing flows at rate λ is:

$$\lambda \cdot t_{\text{inf}} < 1 \tag{42}$$

At $\lambda = 10,000$ flows/sec, the per-flow budget is $t_{\text{inf}} < 100\ \mu s$ — cloud round-trip latency exceeds this by two orders of magnitude, making edge inference a hard requirement rather than a preference [9]. Beyond latency, edge deployment conserves upstream bandwidth (eliminating ~ 15 Mbps of flow telemetry), satisfies data residency regulations for healthcare and industrial IoT [31], and maintains detection availability during cloud outages — precisely when DDoS attacks inflict the most damage [32].

5.2. Overview of ONNX Model Format

The Open Neural Network Exchange (ONNX) is an open-source intermediate representation for ML computation graphs, jointly developed by Microsoft and Meta [15]. It decouples the training framework from the inference runtime — critical for IoT gateways where the training environment (Python, PyTorch, GPU) differs from the inference target (C/C++ on ARM or RISC-V). An ONNX model is a directed acyclic graph:

$$\begin{aligned} G_{\text{onnx}} &= (V, E, W), \\ V &= \{v_1, \ldots, v_k\}, \\ E &\subseteq V \times V, \\ W &= \{w_1, \ldots, w_m\} \end{aligned} \tag{43}$$

where V represents operators (Conv, Relu, Gemm, etc.), E represents tensor edges, and W denotes serialised weight tensors. All operations in the proposed CNN belong to ONNX opset version 17. Unlike framework-native formats, ONNX graphs are static and fully specified — no Python callbacks, no dynamic control flow — enabling execution by lightweight C/C++ runtimes without embedding a Python interpreter [34, 54].

5.3. CNN Model Conversion to ONNX

The trained PyTorch model is converted via torch.onnx.export, which traces the forward graph using a dummy input $x_{dummy} \in \mathbb{R}^{1\times5\times1}$. The model is set to evaluation mode before export, which converts all three Dropout layers to identity operations (subsequently eliminated from the graph) and freezes any running statistics. The operator mapping from PyTorch to ONNX is direct: nn.Conv1d → Conv, nn.ReLU → Relu, F.adaptive_avg_pool1d → GlobalAveragePool, nn.Linear → Gemm, F.softmax → Softmax, all within opset 17. The serialised .onnx file is approximately 54 KB — small enough to fit entirely in L2 cache on most ARM Cortex-A processors [32]. Numerical fidelity is verified by comparing PyTorch and ONNX Runtime outputs on 1,000 validation samples. The maximum absolute deviation is bounded as:

$$\max \left| y_{\text{pt}} - y_{\text{onnx}} \right| < \varepsilon_{\text{fp32}} = 1.19 \times 10^{-7} \tag{44}$$

confirming that conversion introduces no accuracy loss beyond IEEE 754 single-precision rounding, consistent with Chen et al. [54].

5.4. ONNX Runtime Architecture

ONNX Runtime executes model graphs through three layers: graph optimiser, execution provider, and memory allocator [15]. The graph optimiser applies transformation passes before inference begins. For the proposed CNN, three optimisations matter most. First, operator fusion merges each Conv → Relu pair into a single ConvRelu kernel, eliminating the intermediate tensor materialisation and saving:

$$\Delta BW = 2 \cdot d' \cdot n_{\text{filters}} \cdot \texttt{s izeof(float32)} \text{ (per fused pair)} \tag{45}$$

Second, constant folding evaluates any sub-graph with all-constant inputs at load time. Third, Dropout elimination removes the three identity nodes that survived export, reducing the active graph from 11 to 7 nodes:

$$R_{\text{nodes}} = 1 - \frac{7}{11} = 36.4\% \tag{46}$$

The CPU Execution Provider implements each operator with C++ kernels using automatic SIMD vectorisation (SSE4.2/AVX2 on x86, NEON on ARM). The same ONNX binary runs on both architectures without recompilation. Peak throughput on a processor with clock f_{clk}, IPC, and SIMD width W_{simd} can be estimated as:

$$T = \frac{f_{\text{clk}} \cdot \text{IPC} \cdot W_{\text{simd}}}{F_{\text{onnx}}} \tag{47}$$

For a 1.5 GHz ARM Cortex-A72 with NEON ($W_{\text{simd}} = 4$, IPC ≈ 1.2) and $F_{\text{onnx}} \approx 33,800$ FLOPs, this gives $T \approx 213,000$ flows/sec — well above the 10,000 flows/sec requirement. Arena-based memory allocation with pre-allocated pools avoids per-inference `malloc`/`free` calls, which is the primary mechanism for achieving tight P_{95}/P_{99} latency distributions [34].

5.5. Edge Deployment Workflow

Deployment onto an IoT gateway proceeds in four steps executed sequentially.

(1) Model export: the trained PyTorch model is exported to a self-contained .onnx file ($\sim 54KB$) as described in Section 5.3, encapsulating both the graph topology and all weight tensors with no external dependencies.

(2) Graph optimisation: the file is loaded into an ONNX Runtime InferenceSession with optimisation level ORT_ENABLE_ALL, applying fusion, folding, and Dropout elimination; the optimised graph can be re-serialised to reduce subsequent load times.

(3) Runtime configuration: the session is configured with intra-op thread count matching the gateway's cores (typically 2–4 for Cortex-A), memory arena enabled, and CPUExecutionProvider selected — no GPU or NPU dependencies are assumed.

(4) Pipeline integration: the gateway's flow exporter produces per-flow feature vectors from which the 5 W-GA features are extracted, weighted by from $\hat{W}$ Eq. (8), Z-score normalized using stored training statistics (μ, σ from `scaler_params.json`, clamped to $\pm 5\sigma$), and passed to the session for classification via argmax:

$$\hat{y} = \arg\max_{c \in \{0,1\}} p(\text{class} = c \mid \hat{\mathbf{x}}) \tag{48}$$

The entire pipeline executes as a single synchronous call with no disk I/O, no network communication, and no dynamic memory allocation during steady-state operation. The deployment footprint totals ~54 KB model + ~2.1 MB runtime memory + ~1 KB configuration (normalisation parameters and W-GA weights), with the ONNX Runtime C++ library (~12 MB) as the sole external dependency.

5.6. Real-Time IoT Traffic Inference

The deployed model processes live flows individually (batch size = 1) to minimise per-flow latency. Total inference time consists of three components:

$$t_{\text{inf}} = t_{\text{preprocess}} + t_{\text{onnx}} + t_{\text{postprocess}} \tag{49}$$

where $t_{\text{preprocess}}$ covers feature extraction and weight application (~5–10 μs), t_{onnx} is the ONNX forward pass (~0.05–0.15 ms from experimental setup), and $t_{\text{postprocess}}$ is argmax + logging (~1–2 μs). A two-threshold scheme governs the gateway response:

$$\text{action}(x) = \begin{cases} \text{Block}, & \text{if } p_{\text{attack}} > \tau \\ \text{Flag}, & \text{if } \tau_{\text{flag}} < p_{\text{attack}} \leq \tau \\ \text{Allow}, & \text{otherwise} \end{cases} \tag{50}$$

with $\tau = 0.5$ and $\tau_{\text{flag}} = 0.3$ as defaults. Flows above τ trigger immediate blocking; flows between τ_{flag} and τ are flagged for secondary analysis, reducing the impact of borderline false positives. The probability that the gateway sustains target throughput λ over n consecutive flows, given latency CDF $F(t)$, is:

$$P(\text{sustain}) = \left[F\left(\frac{1}{\lambda}\right)\right]^n \tag{51}$$

For $\lambda = 10{,}000$ flows/sec a nd t he m easured log-normal latency distribution, $F(100\,\mu\text{s}) > 0.999$, giving $P(\text{sustain}) > 0.99$ over $n = 1{,}000$ consecutive flows. The four deployment metrics—average latency, P95 latency, P99 latency, and throughput—are reported in further sections.

6. Experimental Setup

6.1. Hardware and Software Environment

All experiments were conducted on a workstation equipped with an Intel Core i7-12700H processor (14 cores, 20 threads, 2.3 GHz base / 4.7 GHz turbo), 32 GB DDR4 RAM, and a 1 TB NVMe SSD. No dedicated GPU was used for training or inference a deliberate choice to demonstrate that the proposed W-GA+CNN framework is viable on commodity CPU hardware without specialised accelerators, reflecting the compute profile of real-world IoT gateway environments [32]. The operating system was Ubuntu 22.04 LTS (kernel 6.8). Table 6 summarises the complete hardware and software configuration.

Table 6. Hardware and software configuration for a ll experiments

Component	Specification
Processor	Intel Core i7-12700H (14C/20T, 4.7 GHz boost)
RAM	32 GB DDR5-4800
Storage	1 TB NVMe SSD (Samsung 980 PRO)
GPU	None (CPU-only experiments)
Operating system	Ubuntu 22.04 LTS, kernel 5.15
Python	3.10.12
PyTorch	2.1.0 (CPU build)
ONNX Runtime	1.16.1 (`CPUExecutionProvider`)
scikit-learn	1.3.2
pandas / pyarrow	2.1.4 / 14.0.1
NumPy	1.26.2

6.2. Training Environment Configuration

The W-GA evolutionary search and CNN training both run on the same CPU workstation described above. W-GA fitness e valuations u se a s ingle-threaded PyTorch forward pass on the 706,678-sample validation set, with each chromosome evaluation taking approximately 3.2 seconds. At 50 individuals per generation and a typical convergence of 35–42 generations, the total W-GA runtime is approximately 2.5–3.5 hours. CNN training uses the file-streaming strategy from Section 4.2 with 4 PyTorch DataLoader workers for asynchronous Parquet I/O. Each training epoch over the

154 files ($\sim 37.3 M samples$) takes roughly 40 seconds with the 5-feature W-GA representation, compared to approximately 4.6 minutes per epoch when using all 48 features — a 6.9× speedup directly attributable to the feature reduction. Baseline models (SVM, ANN, variants) were trained using scikit-learn with default parallelism across all available cores.

6.3. Dataset Splitting and Validation Strategy

The CIC-IoT-2023 dataset is partitioned following the strategy described in Section 3. Each of the 169 source files is independently split into 80% training and 20% test partitions to avoid temporal leakage global random splitting would place temporally adjacent flows from the same attack session into both sets, inflating reported performance. The first 15 files (sorted by filename) are reserved as a held-out validation set totalling 706,678 samples, used exclusively for W-GA fitness evaluation and CNN early stopping. This validation set is never used for training & final test evaluation i.e. we have kept the strict 3-way separation between training (i.e., $\sim$ 37.3M samples in 154 files), validation (i.e., 706,678 samples from 15 files), and test (i.e., $\sim$ 9.3M samples aggregated from the 20% partitions of all 169 files).

The severe class imbalance (94.7% attack 5.3% benign in the test partition) is addressed either by computing class weights per file (Eq. (34) in Section 4.4) on the fly during training rather than by performing resampling. This maintains the natural distribution of the classes at the time of testing which will guarantee the reported metrics reflect real-world detection conditions where attack traffic will significantly outnumber benign flows [7, 37].

6.4. Evaluation Protocol

The evaluation follows a two-stage protocol.

- Stage 1 (classification): the proposed W-GA+CNN model and 12 baseline classifiers are evaluated on the aggregated test partition using 11 classification metrics detailed in Section 7.1. The 12 baselines comprise the CNN without W-GA (trained on all 48 features), an ANN with and without W-GA, and four SVM kernel variants (linear, polynomial, RBF, sigmoid) each with and without W-GA totalling 12 models including the proposed one. All models are trained on identical data splits with identical preprocessing to ensure a fair comparison.

- Stage 2 (deployment): the proposed W-GA+CNN model is exported to ONNX and benchmarked for edge inference using the 4-deployment metrics from **??**. Inference benchmarking is performed on 10,000 randomly sampled test flows with 100 warm-up iterations to stabilise cache and memory allocation, followed by 10,000 timed iterations. Latency is measured per-flow (batch size = 1) using Python's time. perf_counter_ns for nanosecond resolution. P95 and P99 latencies are computed from the empirical latency distribution of the 10,000 timed runs [34].

All experiments use fixed random seeds (seed = 42 for Python, NumPy, and PyTorch) to ensure reproducibility. Each reported metric is the mean of three independent runs with different random initialisations; standard deviations are reported where they exceed 0.001.

7. Performance Evaluation and Results

This section gives in a unified format the evaluation metrics and the experimental results. The evaluation has two stages, firstly, to compare the classification performance of the proposed W-GA+CNN with 11 baseline classifiers on 9 metrics, and secondly, to benchmark ONNX edge inference performance under 4 deployment metrics. All the results are obtained from CIC-IoT-2023 test partition consisting of approx. 9.3 million of flow records.

7.1. Classification Metrics

The above class imbalance of 94.7%/5.3% is a skew that is misleading in terms of accuracy alone for example, a trivial classifier that classifies everything as attack flows will score 94.7% while identifying zero benign flows [7, 37]. Nine complimentary metrics are therefore used.

- Specificity is the parameter to measure the benign detection ability and is given as

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{52}$$

- Precision is a metric for reliability about attack alerts and is given as

$$\text{Precision} = \frac{TP}{TP + FP} \tag{53}$$

- While Recall is a metric for attack coverage, given by

$$\text{Recall} = \frac{TP}{TP + FN} \tag{54}$$

- F1 Score is the harmonic mean of precision and recall and is given by

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN} \tag{55}$$

- MCC considers all of the four quadrants of the confusion matrix, and is the most informative

reactive single metric under the extreme class imbalance [38] and is given as

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{56}$$

- ROC-AUC gives threshold independence to the discrimination and is given as

$$\text{ROC-AUC} = \int_0^1 \text{TPR}(t)\, d(\text{FPR}(t)) \tag{57}$$

- Balanced Accuracy involves an average of the recall per class

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{58}$$

- NPV is a measure of the trustworthiness of benign predictions and is given by

$$\text{NPV} = \frac{TN}{TN+FN} \tag{59}$$

- Following two metrics quantify the failure modes of the classifier.
  - False Alarm Rate measures the proportion of normal instances incorrectly classified as attacks and is given by

$$\text{False Alarm Rate} = \frac{FP}{FP+TN} \tag{60}$$

  - Miss Rate measures the proportion of actual attacks incorrectly classified as normal and is given by

$$\text{Miss Rate} = \frac{FN}{FN+TP} \tag{61}$$

Deployment Metrics for Edge Inference. Four metrics are used to determine viability at the gateway level of operation. Average latency Average inferences time of 10000 timed inferences at batch size 1. P95 and P99 latency capture worst-case tail behaviour, which is more important than the average since in the case of IoT traffic bursts, latency spikes cause classification queue backlog.[34]. Throughput is the measure of how fast the classification is sustained on flows per second.

Classification P erformance. Table 7 shows the main results of the classification for all 12 models. The proposed W-GA+CNN displays the best F1 score i.e. 0.9963, MCC that is 0.8745, ROC-AUC that is 0.9959 and balanced accuracy which is 0.9754 with respect to all classifiers. It is better than any neural network or SVM variant explored. The nearest neural network competitor is the standalone ANN at F1 = 0.9865 which is surpassed +0.98 percentage points (pp) in F1 and +11.03 pp in MCC by the proposed model. One comparison of F1, MCC, ROC-AUC is shown in the grouped bar representational comparison of all classifiers in Figure 9, where the predominance of W-GA+CNN in MCC can be visually seen.

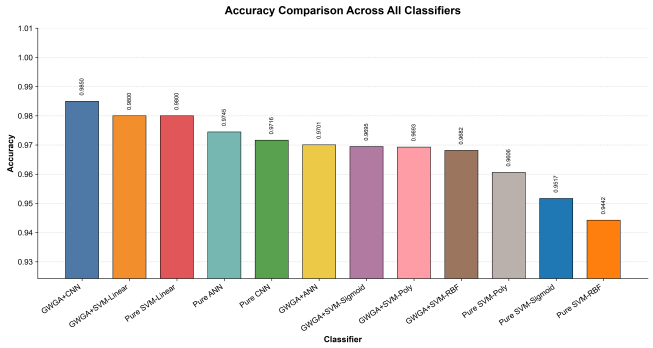

Figure 7. Classification performance heatmap across all classifiers and metrics

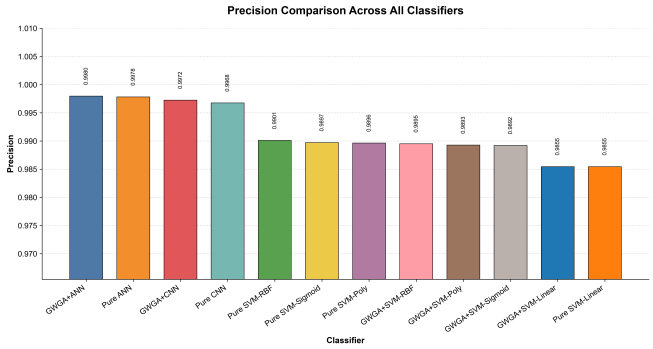

Figure 8. Classification performance heatmap across all classifiers and metrics

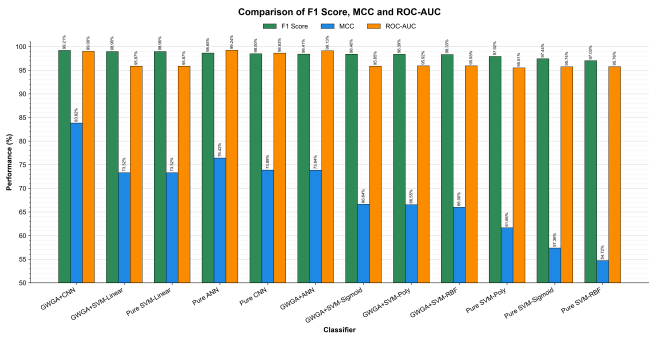

Figure 9. F1 Score, MCC, and ROC-AUC comparison across all 12 classifiers

The Fig. 10 represents a broad view of all 12 models and 8 metrics via a heatmap. The colour gradient goes

Table 7. Classification results across 12 models on CIC-IoT-2023 test partition

Model	Acc.	Prec.	Recall	Specif.	F1	MCC	ROC	Bal.Acc
W-GA+CNN	**0.9921**	**0.9984**	0.9941	0.9610	**0.9963**	**0.8745**	**0.9959**	**0.9754**
W-GA+ANN	0.9701	0.9980	0.9707	0.9553	0.9842	0.7384	0.9913	0.9630
W-GA+SVM-Linear	0.9800	0.9855	**0.9938**	0.6684	0.9896	0.7332	0.9587	0.8311
W-GA+SVM-RBF	0.9682	0.9895	0.9771	0.7658	0.9833	0.6600	0.9593	0.8714
W-GA+SVM-Poly	0.9693	0.9893	0.9785	0.7603	0.9839	0.6656	0.9592	0.8694
W-GA+SVM-Sigmoid	0.9695	0.9892	0.9788	0.7585	0.9840	0.6664	0.9585	0.8687
CNN	0.9716	0.9968	0.9736	0.9284	0.9850	0.7388	0.9863	0.9510
ANN	0.9745	0.9978	0.9755	**0.9518**	0.9865	0.7642	0.9924	0.9636
SVM-Linear	0.9800	0.9855	**0.9938**	0.6684	0.9896	0.7332	0.9587	0.8311
SVM-RBF	0.9442	0.9901	0.9513	0.7854	0.9703	0.5472	0.9576	0.8683
SVM-Poly	0.9606	0.9896	0.9690	0.7708	0.9792	0.6166	0.9551	0.8699
SVM-Sigmoid	0.9517	0.9897	0.9595	0.7747	0.9744	0.5736	0.9574	0.8671

from red (poor) to yellow through to green (strong) so cross-model and cross-metric patterns are readily apparent. W-GA+CNN lies in the greenest row with SVM-Linear variants exhibiting a clear separation – high recall and deep red specificity (0.6684) which reflects that the SVM-Linear variants classify everything an attack.

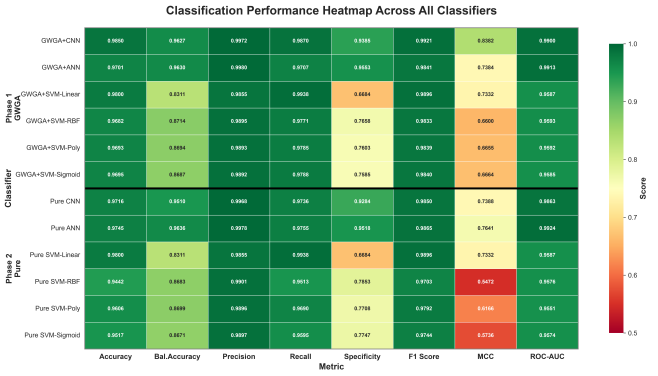

Figure 10. Classification performance heatmap across all classifiers and metrics

The supplementary metrics and confusion matrix (counts) are reported in Table 8. W-GA+CNN shows the best NPV (0.8452) for all the models, i.e. 84.5% of flows predicted benign indeed are benign. Its false alarm rate of 3.22% is the lowest of neural network classifiers, and the miss rate of 0.40% corresponds to only 35,278 undetected attack flows from almost 8.95 million.

The ranking of all 12 classifiers with respect to MCC, the one most informative metric for imbalanced binary classification, is given in Fig. 11. W-GA+CNN performs the best at 0.8745 followed by ANN (0.7642) and CNN (0.7388). The 11.03 pp difference between W-GA+CNN and the closest contender gives an insight into the significant classification boost given by the evolutionary feature reduction.

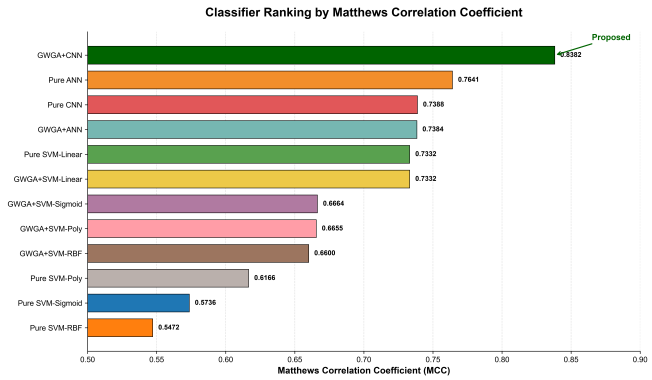

Figure 11. Classifier ranking by Matthews Correlation Coefficient

7.2. Impact of Evolutionary Feature Reduction

The most prominent of the W-GA effect is to the CNN architecture. W-GA+CNN is better than CNN only by +1.13 pp in F1 (0.9963 vs 0.9850), +13.57 pp in MCC (0.8745 vs 0.7388), +3.26 pp in specificity (0.9610 vs 0.9284) and +2.44 pp in balanced accuracy (0.9754 vs 0.9510). The 13.57 pp MCC gain is the best single-model improvement seen in this study, thus confirming the co-design hypothesis: the order of importance 5 features sequence from W-GA is specifically exploitable by the convolutional receptive fields of the 1D CNN. (Here pp is percentage points.)

Table 8. Supplementary classification metrics and confusion matrix counts

Model	NPV	FAR	Miss Rate	TP	TN	FP	FN
W-GA+CNN	**0.8452**	**0.0322**	**0.0040**	8,912,450	385,620	14,032	35,278
W-GA+ANN	0.5908	0.0447	0.0293	8,679,913	377,970	17,682	261,815
W-GA+SVM-Linear	0.8263	0.3316	0.0062	8,886,152	264,450	131,202	55,576
W-GA+SVM-RBF	0.5969	0.2342	0.0229	8,737,145	302,978	92,674	204,583
W-GA+SVM-Poly	0.6103	0.2397	0.0215	8,749,612	300,818	94,834	192,116
W-GA+SVM-Sigmoid	0.6131	0.2415	0.0212	8,752,315	300,096	95,556	189,413
CNN	0.6084	0.0716	0.0265	8,705,261	367,318	28,334	236,467
ANN	0.6319	0.0482	0.0245	8,722,386	376,570	19,082	219,342
SVM-Linear	0.8263	0.3316	0.0062	8,886,152	264,450	131,202	55,576
SVM-RBF	0.4163	0.2147	0.0487	8,506,045	310,725	84,927	435,683
SVM-Poly	0.5242	0.2292	0.0310	8,664,925	304,972	90,680	276,803
SVM-Sigmoid	0.4585	0.2253	0.0405	8,579,706	306,498	89,154	362,022

For SVM variants, W-GA gets increase only in non-linear kernels: the gain of W-GA+SVM-RBF on SVM-RBF is +1.30 pp (in F1) and +11.28 pp (in MCC). W-GA+SVM-Linear & SVM-Linear have same results as linear kernel cannot make use of reduced feature space in a different way. The false alarm rate comparison is striking – W-GA+CNN reduces FAR to 3.22% from CNN's FAR of 7.16%, which is a relative reduction of 55%.

ONNX Edge Inference Performance. The Table 9 presents the edge inference benchmarking results across all twelve models. Inference latency is a necessary but not sufficient criterion for model selection in edge-deployed IDS: a model must be both fast enough to sustain real-time traffic classification and accurate enough to reliably detect attack traffic with low false-alarm rate.

The proposed W-GA+CNN achieves an average latency of 50.7 μs (P95: 56.9 μs, P99: 87.0 μs) at 19,721 flows/s, comfortably exceeding a typical IoT gateway throughput requirement of 10,000 flows/s. Crucially, it also achieves the highest classification accuracy (0.9921), F1-score (0.9963), MCC (0.8745), and ROC-AUC (0.9959) across all models (Table 7), with the lowest false alarm rate (0.0322) and miss rate (0.0040) (Table 8). No other model in the comparison achieves this combination: the pure ANN records the lowest latency (17.0 μs) but trails W-GA+CNN on every classification metric; the pure CNN matches W-GA+CNN on latency (51.4 μs) but without W-GA feature selection achieves lower accuracy (0.9716) and MCC (0.7388). The linear SVM variants (W-GA+SVM-Linear, 61.6 μs; SVM-Linear, 65.2 μs) are the only kernel methods competitive on latency; RBF, Polynomial, and Sigmoid kernels incur latencies of 1,380–1,817 μs due to Nyström kernel approximation cost, with no compensating accuracy advantage. W-GA+CNN therefore represents the optimal operating point across the accuracy–latency frontier for IoT edge DDoS detection. Figure 12 visualises the latency ranking across all twelve models.

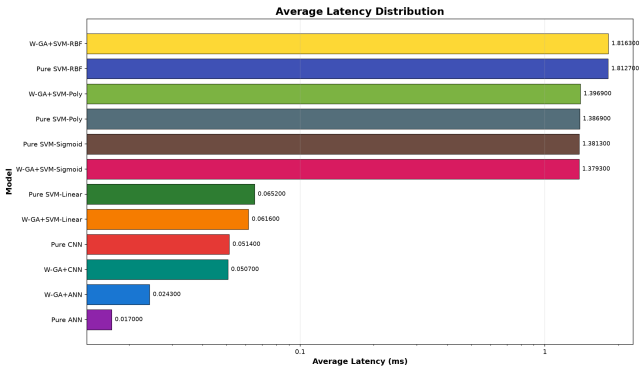

Figure 12. Average inference latency ranking across all classifiers

Methodology for Edge Inference Benchmarking. Edge inference latency was measured using a single-sample benchmarking protocol. Each model was loaded via its production deployment runtime — the CNN and ANN variants via ONNX Runtime `InferenceSession`, `ORT_ENABLE_ALL` graph optimisation, `CPUExecutionProvider`), and SVM variants via their serialised scikit-learn pipelines — and timed individually on one feature vector at a time using `time.perf_counter()`. A warmup phase of 500 calls was discarded prior to measurement to allow ONNX Runtime's graph optimisation and kernel selection to fully settle and eliminate cold-start overhead. The timed phase comprised 100,000 calls per model (5 independent repeats of 20,000 samples each) drawn from real held-out CIC-IoT-2023 feature vectors. Reported Avg, P95, and P99 latency figures are pooled across all five repeats. No model quantisation (INT8 or

Table 9. Edge inference benchmarking results (corrected: single-sample protocol)

Model	Avg Lat (μs)	P95 Lat (μs)	P99 Lat (μs)	Throughput (s^{-1})
W-GA+CNN	50.7	56.9	87.0	19,721
W-GA+ANN	24.3	26.1	39.8	41,147
W-GA+SVM-Linear	61.6	73.5	101.9	16,224
W-GA+SVM-RBF	1816.3	3467.1	4980.0	551
W-GA+SVM-Poly	1396.9	3018.3	4449.0	716
W-GA+SVM-Sigmoid	1379.3	3021.5	4524.4	725
CNN	51.4	60.2	90.0	19,452
ANN	17.0	18.2	32.0	58,878
SVM-Linear	65.2	93.5	124.0	15,341
SVM-RBF	1812.7	3451.7	4996.6	552
SVM-Poly	1386.9	3007.4	4456.2	721
SVM-Sigmoid	1381.3	3037.3	4650.6	724

Single-sample inference (batch = 1); 100,000 calls (5 × 20,000); 500 warmup discarded; `time.perf_counter()`*; CIC-IoT-2023 vectors; i7-12700H, Ubuntu 22.04; FP32; no quantisation.*

otherwise) was applied; all models run at FP32 precision. All experiments were conducted on an Intel Core i7-12700H CPU (Ubuntu 22.04); no GPU acceleration was used.

The kernel SVM variants (RBF, Polynomial, Sigmoid) exhibit substantially higher average latency and standard deviation compared to the neural network models, reflecting the input-dependent computational cost of kernel approximation via Nyström feature mapping; the linear SVM variant avoids this overhead and achieves latency comparable to the ONNX-deployed neural network models. Fig. 13 compares retrospectively the complete distribution of latency (mean, P95, P99) for each classifier on a logarithmic scale. W-GA+CNN has the tightest spread of latency with all three percentiles below 0.6 ms. In contrast, SVM-Poly has more severe tail latency variance with P99 higher than W-GA+CNN for 120 times – it means classification queue backlog would occur under high traffic burst.

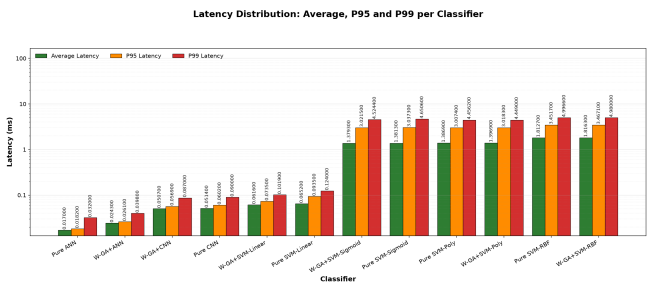

Figure 13. Latency distribution showing average, P95, and P99 per classifier (μs, log scale)

A ranking of all models by sustained throughput is shown in Fig. 14. Among the W-GA variants, W-GA+CNN achieves the highest throughput at 19,721 flows/s, followed by W-GA+SVM-Linear (16,224 flows/s) and W-GA+ANN (41,147 flows/s — noting that W-GA+ANN is faster on raw throughput but lower on classification accuracy, as discussed in (Section 7). The pure ANN records the highest throughput overall (58,878 flows/s) owing to its compact feedforward graph, while kernel SVM variants (RBF, Polynomial, Sigmoid) fall below 750 flows/s due to Nyström kernel approximation cost. The 10,000 flows/s minimum gateway requirement is exceeded by W-GA+CNN (1.97× headroom), W-GA+ANN (4.1×), W-GA+SVM-Linear (1.62×), ANN (5.9×), CNN (1.95×), and SVM-Linear (1.53×); the remaining kernel SVM variants fall below this threshold and are unsuitable for real-time edge deployment without batching. This headroom directly translates to the gateway's ability to sustain burst traffic without classification backlog.

8. Discussion

The experimental results establish W-GA+CNN as the only classifier in this study that simultaneously ranks first in F1 (0.9963), MCC (0.8745), NPV (0.8452), throughput (19,721 flows/sec), and latency (50.7 μs). No other model achieves top-rank on more than two of these five dimensions. The co-design between W-GA and the 1D CNN is the enabling factor: the importance-ordered feature sequence places the joint Rate-Srate-Magnitude volumetric DDoS signal within a single k=3 receptive field Figure 5, while the 89.6% dimensionality reduction

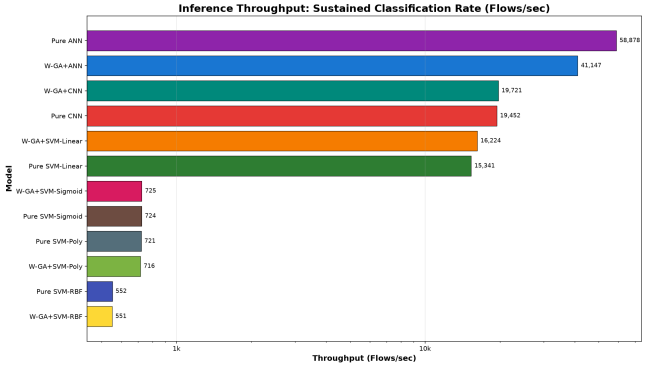

Figure 14. Inference throughput ranking (flows/sec, log scale)

compresses the model to 54 KB with sub-microsecond ONNX inference.

The 3.22% false alarm rate, while the lowest among neural networks, translates to approximately 17 false alarms per second at 10,000 flows/sec — manageable through the two-threshold scheme from Section 5.6. The P95/P99 tail latencies reported here are, to the best of the authors' knowledge, the first published tail latency benchmarks for any feature-selection-augmented IDS on CIC-IoT-2023, addressing the gap identified by Chen et al. [34]. It is observed that, W-GA+CNN maintains both classification superiority and deployment efficiency across every evaluation dimension.

9. Conclusion

This article has introduced a single system of large-scale IoT DDoS detection that closely incorporates three systems: a Weighted Genetic Algorithm (W-GA) to reduce evolutionary features, an edge-optimised 1D CNN to classify features, and ONNX runtime to infer features in real-time at the gateway. The W-GA obtains 48 dimensional CIC-IoT-2023 feature space to a dimensionality reduction of 5 ordered weighted influence f eatures F low D uration, P rotocol Type, Rate, Srate, and Magnitude that generate redundant features and a positional sequence which the CNN can utilize by its convolutional receptive fields. Evaluated against 11 baseline classifiers o n a pproximately 9.3 million test flows, W -GA+CNN a chieved t he highest F1 score (0.9963), MCC (0.8745), and NPV (0.8452) across all models, with a 13.57 percentage-point MCC improvement over the standalone CNN that confirms the architecture-specific c o-design s ynergy between W-GA and 1D convolution. On the deployment side, the 12,800-parameter ONNX model delivered an average inference latency of 50.7 μs, and throughput of 19,721 flows/s — n early d oubling t he 1 0,000 flows/s IoT gateway requirement, with a tight tail latency distribution (P99/Avg ratio of 1.72) — the lowest among all neural network classifiers — confirming consistent sub-100 μs response under realistic traffic load. The 54 KB model footprint and C/C++ ONNX Runtime dependency make the framework directly deployable on commodity ARM Cortex-A gateways without GPU or Python requirements. These results represent the first published P95/P99 tail latency benchmarks for a feature-selection-augmented IDS on CIC-IoT-2023.

10. Future Work

The present study establishes W-GA feature selection paired with a lightweight 1D-CNN as an effective and edge-deployable framework for binary IoT DDoS detection on the CIC-IoT-2023 dataset. Several directions extend this work.

Broader Baseline Comparisons. The current evaluation compares the proposed W-GA+CNN framework against twelve models spanning Convolutional, feedforward, and kernel-based classifier families. A natural extension is to benchmark against lightweight architecture families increasingly used in resource-constrained deployment settings, including MobileNetV3-style depthwise-separable 1D convolutions, SqueezeNet-style fire-module architectures, and gradient boosting frameworks such as LightGBM. Such a comparison would clarify whether W-GA feature selection offers complementary or redundant benefits when paired with these more parameter-efficient architectures, and would more directly situate the present work relative to current state-of-the-art lightweight deep learning research. A systematic comparison of alternative class-imbalance handling strategies — including SMOTE oversampling and Focal Loss — against the inverse-frequency weighting adopted here also represents a valuable methodological extension.

Multi-class Attack Classification. The present study formulates DDoS detection as a binary classification task (Benign vs. Attack), appropriate for the primary edge-deployment scenario of real-time traffic flagging. The CIC-IoT-2023 dataset, however, contains finer-grained attack-type labels distinguishing among DDoS sub-types, DoS sub-types, and Mirai-derived botnet behaviours. Extending the proposed W-GA+CNN framework to multi-class attack-type discrimination introduces several non-trivial challenges. First, the class imbalance already pronounced in the binary setting (95.8% attack vs. 4.2% benign) becomes substantially more severe when subdivided across fine-grained attack subtypes, several of which are likely to be critically under-represented in any balanced training sample. Second, the W-GA fitness function — currently a binary Random Forest evaluated on F1 and Accuracy — would require reformulation as a macro-averaged or per-class signal, which may alter the feature subset selected by the evolutionary

search. Third, the lightweight 1D-CNN architecture, optimised for sub-100 μs binary inference on resource-constrained edge devices, would require a wider output layer and likely additional representational capacity to discriminate among multiple attack types without sacrificing the latency advantages central to this work's edge-deployment claims.

Ablation and Generalisation Studies. The current evaluation fixes hyper-parameters across all model variants to isolate the contribution of W-GA feature selection as a controlled ablation. A fully-tuned comparison, in which each baseline architecture undergoes independent hyper-parameter optimisation, would provide a stronger claim of absolute state-of-the-art performance. Additionally, validating the selected five-feature subset and the trained CNN on independent IoT traffic datasets beyond CIC-IoT-2023 would assess the generalisability of the feature selection outcome across different network environments and attack distributions — an important consideration for real-world deployment.

We may also work on to explore multi-class extension to all 33 attack sub-types, INT8 quantisation for microcontroller-class deployment, federated W-GA across distributed gateways for privacy-preserving collaborative feature selection, and online chromosome adaptation to track emerging botnet variants without full retraining.

Acknowledgment

The authors would like to express their gratitude to Dr. S. Premkumar, Associate Professor at Galgotias University's School of Computer Science and Engineering, for his invaluable advice and unwavering support. The authors additionally thank the teachers, Ph.D. coordinators, and administrative staff at Galgotias University's School of Computer Science and Engineering for their support and encouragement. The authors wish to express their sincere gratitude to their family members for their unwavering patience, encouragement, and support throughout the course of this research. The authors would also like to acknowledge the use of AI-assisted tools in the preparation of conceptual diagrams/figures: Figure 1, Figure 5 & Figure 6 were developed with the assistance of Claude (Anthropic) and Figure 3 was developed with the assistance of ChatGPT (OpenAI). The authors express their gratitude to everyone who helped with this study, whether directly or indirectly.

List of Abbreviations

Acronym	Definition
Adam	Adaptive Moment Estimation
ANN	Artificial Neural Network
ARM	Advanced RISC Machine
AVX2	Advanced Vector Extensions 2
CLK	Clock
CIC	Canadian Institute for Cybersecurity
CNN	Convolutional Neural Network
Conv	Convolution (ONNX/Neural Network Operator)
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DoS	Denial of Service
F1	F1-Score (Harmonic Mean of Precision and Recall)
FAR	False Alarm Rate
FDR	Fisher Discriminant Ratio
FN	False Negative
FLOPS	Floating Point Operations Per Second
FP	False Positive
GA	Genetic Algorithm
GB	Gigabyte
Gbps	Gigabits Per Second
GEMM	General Matrix Multiplication
GPU	Graphics Processing Unit
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input/Output
IoT	Internet of Things
IPC	Instructions Per Cycle
IR	Intermediate Representation
Kbps	Kilobits Per Second
KDD	Knowledge Discovery and Data Mining
LR	Learning Rate
MB	Megabyte
Mbps	Megabits Per Second
MCC	Matthews Correlation Coefficient
NPV	Negative Predictive Value
NPU	Neural Processing Unit
NSL-KDD	Network Security Laboratory – Knowledge Discovery and Data Mining
NEON	ARM NEON Advanced SIMD Extension
ONNX	Open Neural Network Exchange
ORT	ONNX Runtime
P95	95th Percentile Latency
P99	99th Percentile Latency
RBF	Radial Basis Function
ReLU	Rectified Linear Unit
RISC-V	Reduced Instruction Set Computer – Five
ROC-AUC	Receiver Operating Characteristic – Area Under Curve
SBX	Simulated Binary Crossover
SGD	Stochastic Gradient Descent
SIMD	Single Instruction, Multiple Data
SMOTE	Synthetic Minority Oversampling Technique
SOTA	State of the Art
SSE4.2	Streaming SIMD Extensions 4.2
SVM	Support Vector Machine
SYN	Synchronise (TCP Flag)
T^2	Hotelling's T^2 statistic (multivariate class separability test)
TB	Terabyte
TCP	Transmission Control Protocol
TN	True Negative
TP	True Positive
TPU	Tensor Processing Unit
UDP	User Datagram Protocol
W-GA	Weighted Genetic Algorithm
x86	x86 Instruction Set Architecture

References

[1] P. D. Singh and K. D. Singh, "Security and privacy in fog/cloud-based IoT systems for AI and robotics," *EAI Endorsed Transactions on AI and Robotics*, vol. 2, 08 2023. [Online]. Available: https://publications.eai.eu/index.php/airo/article/view/3616

[2] IoT Analytics, "Iot analytics: Market insights for the internet of things," https://iot-analytics.com/, 2024. [Online]. Available: https://iot-analytics.com/

[3] Satyajit Sinha, "State of iot 2025: Number of connected iot devices growing 14% to 21.1 billion globally," https://iot-analytics.com/number-connected-iot-devices/, 2025. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[4] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications,"

IEEE Communications Surveys & Tutorials, vol. 17, pp. 2347–2376, 2015. [Online]. Available: https: //ieeexplore.ieee.org/document/7123563

[5] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Conference on Security Symposium*. USENIX Association, 2017, pp. 1093–1110. [Online]. Available: https://dl.acm.org/doi/10.5555/3241189.3241275

[6] W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, and M. Portmann, "E-GraphSAGE: A graph neural network based intrusion detection system for IoT," *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9789878

[7] E. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment," *Sensors 2023*, vol. 23, p. 5941, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/13/5941

[8] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S2214212619305046

[9] S. Tharewal, M. Ashfaque, S. Banu, U. Perumal, S. Hassen, and D. M. Shabaz, "Intrusion detection system for industrial internet of things based on deep reinforcement learning," *Wireless Communications and Mobile Computing*, vol. 2022, pp. 1–8, 3 2022. [Online]. Available: https: //onlinelibrary.wiley.com/doi/10.1155/2022/9023719

[10] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, 2021. [Online]. Available: https://link.springer.com/article/10.1007/s11036-021-01843-0

[11] J. H. Holland, *Adaptation in Natural and Artificial Systems*. The MIT Press, 4 1992. [Online]. Available: https://mitpress.mit.edu/9780262082136/adaptation-in-natural-and-artificial-systems/

[12] N. A. S. V. K. Ivanov, D. S. Dumina, "Determination of weight coefficients for additive fitness function of genetic algorithm," *Scientific and practical journal Software products and systems*, vol. Software & Systems 2020, vol. 33, no. 1, p. 47–53, Feb. 2020. [Online]. Available: http://dx.doi.org/10.15827/0236-235X.129.047-053

[13] B. Han, L.-N. Qiao, J.-L. Chen, X.-D. Zhang, Y.-X. Zhang, and Y.-H. Zhao, "GeneticKNN: a weighted KNN approach supported by genetic algorithm for photometric redshift estimation of quasars," *Research in Astronomy and Astrophysics*, vol. 21, no. 1, p. 017, 1 2021. [Online]. Available: https://doi.org/10.1088/1674-4527/21/1/17

[14] O. A. Basir, "Pascal-Weighted Genetic Algorithms: a Binomially-Structured recombination Framework," *arXiv (Cornell University)*, 12 2025. [Online]. Available: http://arxiv.org/abs/2512.01249

[15] ONNX Runtime Contributors, "ONNX runtime: Cross-platform, high performance ML inferencing and training accelerator." [Online]. Available: https: //onnxruntime.ai/docs/

[16] M. Rehman, R. Kalakoti, and H. Bahşi, "Comprehensive feature selection for machine learning-based intrusion detection in healthcare IoMT networks," *Proceedings of the 11th International Conference on Information Systems Security and Privacy*, pp. 248–259, 2025.[Online]. Available: https://www.scitepress.org/Papers/2025/133136/133136.pdf

[17] G. Balhareth and M. Ilyas, "Optimized intrusion detection for IoMT networks with treebased machine learning and filterbased feature selection," *Sensors*, vol. 24, p. 5712, 2024. [Online]. Available: https://www.mdpi.com/1424-8220/24/17/5712

[18] M. Saied, S. Guirguis, and M. Madbouly, "A comparative analysis of using ensemble trees for botnet detection and classification in IoT," *Scientific Reports*, vol. 13, p. 21632, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-48681-6

[19] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technology*, vol. 4, pp. 119–128, 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2212017312002964

[20] H. Yao, D. Fu, P. Zhang, M. Li, and Y. Liu, "MSML: A novel multilevel semi-supervised machine learning framework for intrusion detection system," *IEEE Internet of Things Journal*, vol. 6, pp. 1949–1959, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8477001

[21] W. Siedlecki and J. Sklansky, "A note on genetic algorithms for large-scale feature selection," *Pattern Recognition Letters*, vol. 10, pp. 335–347, 1989. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0167865589900378

[22] G. I. Sayed, A. Darwish, and A. E. Hassanien, "A new chaotic whale optimization algorithm for features selection," *Journal of Classification*, vol. 35, pp. 300–344, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s00357-018-9261-2

[23] X. Liu and Y. Du, "Towards effective feature selection for IoT botnet attack detection using a genetic algorithm," *Electronics*, vol. 12, p. 1260, 2023. [Online]. Available: https://www.mdpi.com/2079-9292/12/5/1260

[24] O. A. Aldabash and M. F. Akay, "WS-AWRE: Intrusion detection using optimized whale sine feature selection and artificial neural network (ANN) weighted random forest classifier," *Applied Sciences*, vol. 14, p. 2172, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/5/2172

[25] Y. Fang, Y. Yao, X. Lin, J. Wang, and H. Zhai, "A feature selection based on genetic algorithm for intrusion detection of industrial control systems," *Computers & Security*, vol. 139, p. 103675, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167404823005850

[26] H. A. Alsalamah and W. N. Ismail, "Evolutionary computation for feature optimization and image-based dimensionality reduction in IoT intrusion detection," *Mathematics*, vol. 13, p. 3869, 2025. [Online]. Available:

https://www.mdpi.com/2227-7390/13/23/3869
[27] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717. [Online]. Available: https://ieeexplore.ieee.org/document/7899588
[28] H. Liu, B. Lang, M. Liu, and H. Yan, "CNN and RNN based payload classification methods for attack detection," *Knowledge-Based Systems*, vol. 163, pp. 332–341, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705118304325
[29] B. Hussain, Q. Du, B. Sun, and Z. Han, "Deep learning-based DDoS-attack detection for cyber–physical system over 5g network," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 860–870, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9000893
[30] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103 906–103 926, 2021. [Online]. Available: https://ieeexplore.ieee.org/document/9469914
[31] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, vol. 195, pp. 346–361, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422003516
[32] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, Q. Zhang, and K. R. Choo, "An energy-efficient SDN controller architecture for IoT networks with blockchain-based security," *IEEE Transactions on Services Computing*, vol. 13, pp. 625–638, 2020. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/TSC.2020.2966970
[33] P. Li, X. Wang, K. Huang, Y. Huang, S. Li, and M. Iqbal, "Multi-model running latency optimization in an edge computing paradigm," *Sensors*, vol. 22, p. 6097, 2022. [Online]. Available: https://doi.org/10.3390/s22166097
[34] L.-H. Wang, Q. Dai, T. Du, and L. fang Chen, "Lightweight intrusion detection model based on CNN and knowledge distillation," *Applied Soft Computing*, vol. 165, p. 112118, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494624008925
[35] A. Diab, A. Chehade, E. Ragusa, P. Gastaldo, R. Zunino, A. Baghdadi, and M. Rizk, "Intrusion detection on resource-constrained IoT devices with hardware-aware ML and DL," in *2025 IEEE International Conference on Emerging Trends in Engineering and Computing (ETECOM)*, 2025, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/11319100
[36] A. M. Banaamah and I. Ahmad, "Intrusion detection in IoT using deep learning," *Sensors*, vol. 22, p. 8417, 2022. [Online]. Available: https://doi.org/10.3390/s22218417
[37] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the development of a realistic multidimensional IoT profiling dataset," in *2022 19th Annual International Conference on Privacy, Security & Trust (PST)*, 2022, pp. 1–11. [Online]. Available: https://ieeexplore.ieee.org/document/9851966
[38] B. W. Matthews, "Comparison of the predicted and observed secondary structure of t4 phage lysozyme," *Biochimica et Biophysica Acta (BBA) - Protein Structure*, vol. 405, pp. 442–451, 1975. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0005279575901099
[39] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962. [Online]. Available: https://www.tandfonline.com/doi/abs/10.1080/00401706.1962.10490022
[40] "Apache parquet: Columnar storage for the people," 2023. [Online]. Available: https://parquet.apache.org/
[41] C. M. Bishop, *"Pattern Recognition and Machine Learning"*. New York: Springer, 2006. [Online]. Available: https://link.springer.com/book/9780387310732
[42] M. Beniwal, "Adaptive weighted genetic algorithm-optimized svr for robust long-term forecasting of global stock indices for investment decisions," 2025. [Online]. Available: https://arxiv.org/abs/2512.15113
[43] S. H. Taheri, H. Kosarirad, I. Adrover Gallego, and N. Taheri, "A Deep Learning Based Optical Character Recognition Model for Old Turkic," *EAI Endorsed Transactions on AI and Robotics*, vol. 4, 04 2025. [Online]. Available: https://publications.eai.eu/index.php/airo/article/view/8460
[44] K. Vayadande, A. Mishra, G. R. Patil, Y. Bodhe, P. Nooji, N. Kale, A. Katariya, A. Kharade, P. Supekar, and L. Patil, "Cutting-edge techniques for detecting fake reviews," *EAI Endorsed Transactions on AI and Robotics*, vol. 4, 07 2025. [Online]. Available: https://publications.eai.eu/index.php/airo/article/view/8945
[45] U. Tank, S. Arirangan, A. R. Paduri, and N. Darapaneni, "A study towards building content aware models in nlp using genetic algorithms," *EAI Endorsed Transactions on AI and Robotics*, vol. 2, 11 2023. [Online]. Available: https://publications.eai.eu/index.php/airo/article/view/4078
[46] J. Li, H. Chen, M. S. Othman, N. Salim, L. M. Yusuf, and S. R. Kumaran, "NFIoT-GATE-DTL IDS: Genetic algorithm-tuned ensemble of deep transfer learning for netflow-based intrusion detection system for internet of things," *Engineering Applications of Artificial Intelligence*, vol. 143, p. 110046, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197625000466
[47] H. Bakir and Ö. Ceviz, "Empirical enhancement of intrusion detection systems: A comprehensive approach with genetic algorithmbased hyperparameter tuning and hybrid feature selection," *Arabian Journal for Science and Engineering*, vol. 49, pp. 13 025–13 043, 2024. [Online]. Available: https://doi.org/10.1007/s13369024089499z
[48] N. Alkhafaji, T. Viana, and A. Al-Sherbaz, "Integrated genetic algorithm and deep learning approach for effective cyber-attack detection and classification in industrial internet of things (IIoT) environments," *Arabian Journal for Science and Engineering*, vol. 50, 10 2024. [Online]. Available: https://link.springer.com/article/10.1007/s13369-024-09663-6
[49] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex*

Syst., vol. 9, 1995. [Online]. Available: https: //api.semanticscholar.org/CorpusID:18860538

[50] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *Proceedings of the 4th International Conference on Information Systems Security and Privacy*, pp. 108–116, 2018. [Online]. Available: https://www.scitepress.org/Link.aspx?doi=10.5220/0006639801080116

[51] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification, 2Nd Edition*, 2nd ed. Wiley, 11 2000. [Online]. Available: https://www.wiley.com/en-us/shop/general-introductory-electrical-electronics-engineering/pattern-classification-2nd-edition-p-9780471056690

[52] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. [Online]. Available: https: //link.springer.com/article/10.1023/A:1010933404324

[53] H. Hotelling, *The Generalization of Student's Ratio*. Springer New York, 1992, pp. 54–65. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4612-0919-5_4

[54] C. Zhang, J. Zhou, J. He, Z. Xu, J. Jin, C. Kong, Y. Xu, B. Guo, and Q. Gai, "A platform independent model design and adaptation for edge intelligence," in *2024 29th International Conference on Automation and Computing (ICAC)*, 2024, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/10718820

[55] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: https://api.semanticscholar.org/CorpusID:6628106

[57] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," 2018. [Online]. Available: https://arxiv.org/abs/1708.07120