DOI: 10.2478/ias-2026-0019 • JIAS • 21(6) • 2026 • 378–393

Article

A Hybrid Z-Isomorphic GNN Framework for Robust DDoS Attack Detection in Software-Defined Networks

Zahirabbas J. Mulani [1,2,*], Suhasini Vijaykumar [1] and Priya Chandran [1]

1 Bharati Vidyapeeth's Institute of Management & Information Technology, Navi Mumbai-4000614,India; suhasini.kottur12@gmail.com (S.V.); priyaci2005@gmail.com (P.C.).
2 SVKM's Mithibai College of Arts, Chauhan Institute of Science and Amrutben Jivanlal College of Commerce and Economics, Mumbai-400056, India
* Correspondence author: zahirabbas.mulani@mithibai.ac.in

Abstract: Although SDN provides a programmable, centrally managed framework for modern networks, that same centralization leaves it exposed to attacks such as Distributed Denial of Service (DDoS). This paper proposes an intrusion detection framework that couples Z-Isomorphic Sigmoid Graph Neural Networks (ZIS-GNN) with Bonobo-Optimization-based (EKPC-BOA) feature selection. The sigmoid-based activation strengthens the graph representation relative to conventional GNNs, capturing complex traffic patterns more faithfully, while the hybrid selector – combining the Bonobo Optimization Algorithm with an entropy score and Pearson correlation – distils the most informative features from the traffic data and thereby improves both efficiency and accuracy. Experiments demonstrate that the proposed ZIS-GNN+EKPC-BOA model attains an accuracy of 97.36%, a precision of 97.37%, and an F1-score of 97.58%, outperforming baseline models such as DNN (89.60%), LSTM (91.68%), BiLSTM (93.77%), and GNN (95.86%), as well as the standard graph baselines GCN (96.18%) and the attention-based GAT (96.58%). The results show the effectiveness of combining graph-based learning with hybrid feature selection for intrusion detection in SDN.

Keywords: SDN; DDoS Detection; ZIS-GNN; Intrusion Detection System; Feature Optimization; Bonobo Optimization Algorithm.

1. Introduction

Software-defined networking (SDN) reshapes how networks are operated by decoupling the control logic from the forwarding hardware. As illustrated in Figure 1, a logically centralized control plane decides how packets traverse the network while the data plane merely forwards them [1]. This separation makes operation far more programmable and responsive, yet the same centralization widens the attack surface: SDN deployments can be targeted through side channels, traffic-monitoring abuse, Address Resolution Protocol (ARP) spoofing, and Denial-of-Service (DoS) campaigns. Detecting such intrusions early is therefore essential. An SDN-based intrusion detection system exploits this centralized vantage point to flag malicious activity, unauthorized access, and anomalous behaviour by continuously inspecting traffic for suspicious patterns. The difficulty is that classical detectors built on fixed rule sets age poorly against threats that mutate to evade known signatures. Operational systems generally fall into two families: signature-based detection, which matches traffic against a catalogue of known attack patterns, and anomaly-based detection, which models normal behaviour and reports deviations from it. By concentrating network state in one place, SDN lets these detectors observe the network globally and respond to emerging incidents far more flexibly than traditional distributed setups. Graph Neural Networks (GNNs) are neural architectures tailored to graph-structured inputs; through iterative message passing they aggregate information from neighbouring nodes and edges to learn expressive representations [2]. They have proven effective for node-level classification, link prediction and whole-graph classification, with applications spanning social networks, molecular modelling and bioinformatics. The discriminative power of standard variants such as Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) is, however, upper-bounded by the Weisfeiler–Lehman (WL) isomorphism test: because they rely on piecewise-linear activations like ReLU when summarising a neighbourhood, certain non-isomorphic graphs collapse to identical embeddings and cannot be told apart [3].

© 2026 Zahirabbas J. Mulani et al., published by Cerebration Science Publishing Co., Limited. This work is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0

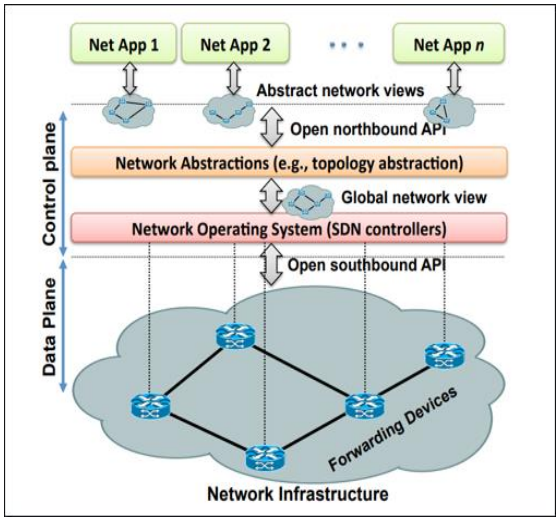

Figure 1. SDN architecture and its fundamental abstractions [4].

Z-Isomorphic Sigmoid GNNs (ZIS-GNNs) extend GNNs by adopting a sigmoid-based activation aimed specifically at this isomorphism limitation. Mapping inputs into a bounded range through the sigmoid provides a non-linearity well suited to probabilistic outputs and improves interpretability in binary classification. The "Z-isomorphic" view holds that, much like Graph Isomorphism Networks (GINs), a non-polynomial activation such as the sigmoid can recognise non-isomorphic graphs that piecewise-polynomial functions (e.g., ReLU) fail to separate. Such models tend to generalise better and are more expressive, especially on complex topologies such as heterogeneous graphs or trees. We term this improved ability to resolve non-isomorphic graphs (trees in particular), achieved by exploiting the transcendental nature of the sigmoid, the Z-Isomorphic property. The sigmoid allows ZIS-GNNs to distinguish non-isomorphic trees in fewer layers than piecewise-polynomial activations such as ReLU. For two non-isomorphic trees of depth two, the root node's representation is (Equation (1)):

$$h_{root}^{(1)} = \sigma\left(W \cdot \sum_{v \in children} \quad x_v + b\right) \tag{1}$$

The transcendental sigmoid ensures distinct outputs for distinct subtree sums, as supported by the Lindemann-Weierstrass theorem [5]. For graph classification, node representations are pooled, as given in Equation (2):

$$h_G = Pool(\{h_u^{(L)} \mid u \in V\}) \tag{2}$$

where $Pool$ is typically a sum or mean. The graph output is given by Equation (3):

$$y = \sigma(W_{out} \cdot h_G + b_{out}) \tag{3}$$

yielding a probability for binary classification.

The proposed work focuses on intrusion detection in software-defined networking (SDN) environments by integrating a Z-isomorphic sigmoid graph neural network (ZIS-GNN) with a hybrid feature selection method. The main contributions of this work are as follows:

- Propose the Z-Isomorphic Sigmoid Graph Neural Network (ZIS-GNN), a sigmoid-activated GNN tailored to DDoS detection, which raises accuracy from 95.86% (plain GNN) to 97.36% – about a 1.5 percentage-point gain on identical data.
- Employ Karl Pearson's entropy-based Bonobo Optimization Algorithm (EKPC-BOA) to select the most informative features, improving classification accuracy along with convergence speed and fitness.
- Assess the proposed strategy against existing deep learning and graph models, including DNN, LSTM, BiLSTM, GNN, and the standard graph baselines GCN (96.18%) and the attention-based GAT (96.58%), using assessment measurements that include precision, recall, F1 score, accuracy, sensitivity, and specificity; the proposed ZIS-GNN (97.36%) outperforms both GCN and GAT, showing that the centred-sigmoid activation adds a further gain over convolution- and attention-based aggregation.

What is new in this work, compared with standard graph models, is the *combination* of two elements rather than any single one: (i) the entropy- and Karl-Pearson-correlation-guided Bonobo Optimization (EKPC-BOA) that selects a compact 14-feature subset, and (ii) the Z-Isomorphic Sigmoid activation within a sum-aggregation GNN. Unlike GCN (spectral mean aggregation), GAT (attention weighting), GraphSAGE (sampled neighbour

aggregation) and GIN (sum aggregation with an MLP), the proposed ZIS-GNN couples sum aggregation with a single centred-sigmoid activation and pairs it with optimization-based feature selection tailored to SDN flow data; the ablation (Section 4.3) isolates the contribution of each part rather than asserting it. Relative to existing SDN-DDoS detectors that mostly use sequence or convolutional models, the contribution is this graph-plus-feature-selection design and its controlled evaluation on CIC-DDoS2019.The remainder of the paper is organized as follows: a review of the literature, limitations of existing SDN-based IDS approaches, the proposed methodology, results, and discussion. We have concluded with an efficient approach to meet the above objective.

2. Related Work

Hadem et al. [6] built an SDN-based IDS that combines support vector machines (SVMs) with selective logging for IP traceback: the SVM separates traffic into normal and anomalous groups, while traceback localizes the true source of attack packets, together improving the tracking of malicious activity. For low-rate DDoS (LR-DDoS) in SDN, Perez-Diaz et al. [7] proposed a flexible, integrated architecture that evaluates several machine-learning classifiers – J48, Random Tree, REP Tree, Random Forest, Multi-Layer Perceptron (MLP) and SVM – and reported a higher LR-DDoS detection rate. Targeting OpenStack clouds, Krishnan et al. [8] fused SDN, network function virtualization (NFV) and ML/AI so that services are virtualized for scalable defence and faults or degradation are handled in real time, improving administration efficiency and security. Bhardwaj et al. [9] addressed intrinsic SDN weaknesses with an IDS framework for Internet of Things (IoT) security, and further SDN-for-IoT studies [10, 11] reported higher accuracy and lower false-positive rates for SDN-based IoT protection. A related direction reduces the control-plane overhead of SDN intrusion detection, easing controller load to make such systems more deployable [12]. Supervised learning approaches for detecting flooding-based DDoS attacks have been widely studied in [13], focusing on accurately distinguishing malicious traffic from legitimate network activity in SDNs. To further increase the detection performance, [14] applied a genetic algorithm for feature optimization, which improved both accuracy and efficiency. An ensemble method, RFAODE [15], fuses multiple classifiers and outperforms single-model detectors, while work on building and classifying comprehensive DDoS datasets [16, 17] has strengthened model training and raised accuracy. Deep learning has likewise been adopted widely – for instance CNN-based detection using geometric descriptors [18] and LSTM-based time-series analysis [19] – with strong results on complex, evolving DDoS attacks. Broader surveys [20] review machine- and deep-learning approaches for SDN intrusion detection and observe that their effectiveness varies with the attack type, while a detailed study of SDN security [21] catalogues vulnerabilities at each architectural tier together with the corresponding countermeasures. Closer to our setting, [22] casts flow records as a graph and shows that a topology- and edge-aware GNN (E-GraphSAGE) surpasses conventional detectors on several NIDS benchmarks, confirming the value of modelling inter-flow relationships rather than treating each flow in isolation. Although existing research on DDoS detection offers useful insights, it also suffers from several recurring drawbacks. Many of these works rely on generated datasets or existing models, yet their applicability across diverse organizational contexts may be limited, as they might not fully capture the variety of real-world attacks. Furthermore, some methods use deep learning models like CNN or LSTM that can be complex as well as computationally demanding; this can make them less practical for real-time detection in high-throughput systems [23, 24]. Moreover, these models tend to overfit when the training data is not sufficiently diverse or representative. Although several studies provide in-depth analysis, their conclusions may not fully reflect the practical performance of the approaches, since they sometimes lack experimental testing or runtime evaluation. In addition, some works, especially those investigating more recent methods such as Graph Neural Networks (GNNs), face challenges related to scalability and computational cost, making them difficult to deploy at scale. Critical advances have been made in intrusion detection, but there is still a need for more versatile and experimentally validated methods that can handle the evolving nature of cyber threats. More recent CIC-DDoS2019 and SDN studies report strong results with hybrid and attention-based deep models – for example, the CNN-BiLSTM with hybrid feature selection of [23], the CNN-with-regularization SDN IDS of [16], and the genetic-algorithm feature-optimization IDS of [14] – yet these mostly treat each flow in isolation and rarely combine optimization-based feature selection with a graph model. The precise gap addressed here is therefore not feature selection, graph modelling or activation design alone, but their *combination*: an entropy–Pearson Bonobo-optimized feature subset fed to a sum-aggregation graph network with a centred-sigmoid activation, evaluated under one controlled pipeline against deep and graph baselines on CIC-DDoS2019.

3. Methodology

The DDOS attack detection phase, as shown in Figure 2, after selecting the dataset, will start with the preprocessing steps, such as data deduplication, which removes repeated data; numeralization, which converts string data into numbers; and normalization, which scales the numerical data to the symmetric range $[-1,1]$ using a MinMax scaler, consistent with the proposed activation. After that, the feature extraction will be done. Next, the optimal features will be selected using the Entropy and Karl Pearson's correlation-guided Bonobo Optimization

Algorithm (EKPC-BOA), built on the Bonobo Optimizer of [25], to improve the classification accuracy. For ease of reference, the mathematical symbols and abbreviations used throughout the paper are summarised in Table 1; each is also defined at its first appearance in the text. For a standard GNN message passing, we define a graph $G = (V, E)$ with node features x_u for $u \in V$. GNNs update node representations via Equation (4):

$$h_u^{(l+1)} = \phi\left(x_u, \oplus_{v \in N(u)} \psi(x_u, x_v, e_{uv})\right) \tag{4}$$

where $N(u)$ is the neighbor set, ψ is the message function, $\oplus$ is aggregation (e.g., sum), and ϕ is an update function (e.g., ReLU) [26].

GNNs model graph-structured data for problems such as node and graph classification, with uses in bioinformatics and social networks [27]. Standard GNNs adopt activations like ReLU, whose expressiveness is capped by the WL test. ZIS-GNNs instead pair sigmoid activations with isomorphism-aware aggregation to better handle non-isomorphic graphs, applying a sigmoid in the update step, as given in Equation (5):

$$h_u^{(l+1)} = \sigma\left(W \cdot \oplus_{v \in N(u)} \psi(x_u, x_v, e_{uv}) + b\right) \ (5) \tag{5}$$

The sigmoid's derivative, used in backpropagation, is given by Equation (6):

$$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z)) \tag{6}$$

For the training loss L, the gradient is given by Equation (7):

$$\frac{\partial L}{\partial z_u} = \frac{\partial L}{\partial h_u^{(l+1)}} \cdot \sigma(z_u) \cdot \left(1 - \sigma(z_u)\right) \tag{7}$$

Inspired by Graph Isomorphism Networks [3], ZIS-GNNs use sum aggregation [5], as defined in Equation (8):

$$h_u^{(l+1)} = \sigma\left(W \cdot \left(x_u + \sum_{v \in N(u)} \quad h_v^{(l)}\right)\right) \tag{8}$$

Node embeddings are pooled and classified, as given in Equation (9):

$$y = \sigma\left(W_{out} \cdot Pool(\{h_u^{(L)} \mid u \in V\})\right) \tag{9}$$

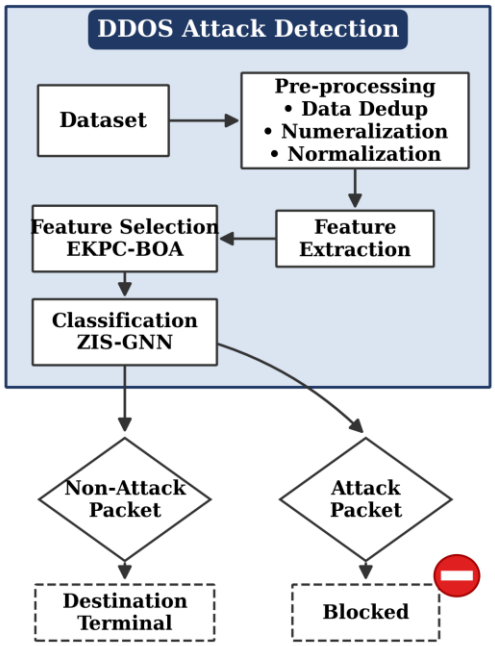

Figure 2. Framework for Proposed System.

Table 1. Summary of notation used throughout the paper.

Symbol	Description
$G = (V, E)$	Flow graph with node set V and edge set E
u, v	Nodes, each representing a network flow
x_u	Input feature vector of node u
$h_u^{(l)}$	Hidden representation of node u at layer l
$N(u)$	Neighbourhood (adjacent flows) of node u
$\oplus$	Permutation-invariant aggregation (sum)
ψ, ϕ	Message and update functions
W, b	Learnable weight matrix and bias
$\sigma(\cdot)$	Z-IsoSigmoid activation $\frac{1}{1+e^{-x}} - \frac{1}{2}$
h_G	Graph-level embedding
$Pool(\cdot)$	Readout (mean / sum pooling)
A, $\hat{A}$, D, I	Adjacency, normalised adjacency, degree, identity
k	Neighbours in the kNN graph ($k = 10$)
K	Number of selected features ($K = 14$)
d	Feature / embedding dimension
L	Training loss
bp	Breeding probability in EKPC-BOA
g_best	Global best solution in EKPC-BOA
$size_b$, $size_w$	Elite and worker group sizes

3.1. CIC-DDoS2019 dataset

CIC-DDoS2019 is a widely used benchmark released by the Canadian Institute for Cybersecurity (CIC) for studying Distributed Denial of Service (DDoS) detection. It was collected in a realistic testbed built from servers, firewalls and IoT devices, and contains both reflection-based and volumetric attack traffic recorded alongside benign activity over an extended period. For every flow, the CICFlowMeter tool exports a rich set of descriptors such as packet sizes, flow durations and header-level statistics. Because it offers a large, consistently labelled mix of attack and normal samples, the dataset is well suited for training and validating intrusion-detection and prevention models against contemporary DDoS threats.

For reproducibility, the labelled flow records used in this work comprise 431,371 flows: 333,540 attack (77.3%) and 97,831 benign (22.7%), giving a class-imbalance ratio of about 3.41:1. The attack traffic spans the reflection- and exploitation-based categories of the dataset (e.g., DrDoS_NTP, TFTP, SYN, UDP, MSSQL, DrDoS_DNS, SNMP, LDAP, NetBIOS, Portmap and WebDDoS). Each flow is described by the CICFlowMeter attributes; after deduplication, string encoding and MinMax scaling, the EKPC-BOA stage reduces these to the 14 selected features. The data is partitioned 80:20 into training and testing sets ($\approx 345{,}097$ training and 86,274 testing flows), with the split performed at the flow level so that no flow appears in both partitions, preventing flow-level leakage. Table 2 summarizes these statistics.

Table 2. CIC-DDoS2019 dataset statistics used in this work.

Property	Value
Total labelled flows	431,371
Attack flows	333,540 (77.3%)
Benign flows	97,831 (22.7%)
Class-imbalance ratio	3.41:1
Raw CICFlowMeter features	80+
Features after EKPC-BOA	14
Train / test split	80:20
Training / testing flows	345,097 / 86,274

3.2. Preprocessing Pipeline

The pipeline starts from the raw flow logs, from which network attributes such as source/destination IP addresses, ports and protocols are extracted. A deduplication step then discards repeated flow entries so that each record corresponds to a distinct network event and redundancy is reduced. Categorical fields such as protocol type and IP-related values are converted to numeric form through string encoding to make them model-compatible, and every numeric column is rescaled to the range $[-1,1]$ with a MinMax scaler to ensure uniform feature ranges and faster convergence.

3.3. EKPC-BOA Feature Selection

Feature Selection Process: **Input:** High-dimensional feature matrix X (80+ features) **Process:**

- Entropy-based scoring for each feature
- Computation of Karl Pearson correlation matrix
- EKPC-BOA optimization (refer to Algorithm 1)

Output: Optimal feature subset (top-K, with $K = 14$ features) This EKPC-BOA algorithm executes in three phases: calculating the Entropy, removing the redundant features, and updating the data to optimize the features. The pseudocode shown here describes a population-based metaheuristic approach in which the solutions evolve over epochs. It starts with initializing the random population, then sorting it by fitness, and tracks the best global solution using exploitation via exponential distribution, which updates with probability bp. Its key components are: population handling, which divides the population into an elite group for local search, of size $size_b = \frac{pop\_size}{5}$, and a set of workers for global diversity; an update rule that promotes convergence, $(1-\alpha)\,pos + e\alpha(g_{best} - pos)$, applied when $random < bp$; and a greedy fitness check that accepts any improvement, resampling the population each epoch as $pos + e \cdot rand()(pos_{t_1} - pos_{t_2})$.

Algorithm 1. Entropy Karl Pearson's Correlation-based Bonobo Optimization Algorithm (EKPC-BOA).

```
Require: Problem parameters, epoch, pop_size, bp
Ensure:  Global best fitness, loss history
Initialize population pop of size pop_size with random solutions
Sort pop by fitness in ascending order
g_best <- best solution in pop
size_b <- floor(pop_size / 5)
size_w <- pop_size - size_b
Initialize loss_train <- empty list
for ep = 1 to epoch do
    for i = 1 to pop_size do
        temp <- pop[i].position
        if i <= size_b then
            if random() < bp then
                alpha <- random()
                temp <- (1 - alpha) * pop[i].position
                        + e * alpha * (g_best.position - pop[i].position)
            end if
        else
            Select random indices t1, t2 in {size_b, ..., pop_size}
            temp <- pop[i].position
                    + e * random() * (pop[t1].position - pop[t2].position)
        end if
        fit <- Calculate the fitness of temp
        if fit < pop[i].fitness then
            pop[i] <- (temp, fit)
        end if
    end for
    Sort pop by fitness in ascending order
```

```
    current_best <- best solution in pop
    if current_best.fitness < g_best.fitness then
        g_best <- current_best
    end if
    Append g_best.fitness to loss_train
    if print_train then
        Print: "Iteration: ep, Best fitness: g_best.fitness"
    end if
end for
return g_best.fitness, loss_train
```

Algorithm 2. Proposed Z-Isomorphic Graph Neural Network (ZIS-GNN).

```
Input:  Network flow dataset D, model parameters
Output: Trained ZISGNN model, evaluation metrics

function TRAINING(D)
    Load dataset from CSV file
    Construct graph-based feature representations from the
      processed network flow records
    Scale features with MinMaxScaler to range [-1, 1]
    Split dataset into training and testing sets (80:20)
    if model does not exist then
        Define custom ZISGNN layer with weight matrix W
        Define activation function:
          Z_IsoSigmoid(x) = 1 / (1 + exp(-x)) - 0.5
        Build neural network with:
            Input layer: dimension = feature count
            Graph convolutional layer using ZISGNN
            Dense layer with Z_IsoSigmoid activation
        Adam optimizer and categorical cross-entropy loss compilation
        Train the model using the specified batch size and
         number of epochs
        Save trained model
    end if
end function
function TESTING( )
    Measure confusion matrix CM = [ TP  FP ; FN  TN ]
    Calculate metrics:
        Precision = TP / (TP + FP)
        Recall    = TP / (TP + FN)
        F1        = 2 * Precision * Recall / (Precision + Recall)
        Accuracy  = (TP + TN) / (TP + TN + FP + FN)
    Save all computed metrics in config variables
    Plot and save the confusion matrix
end function
```

3.4. ZIS-GNN Architecture

Network flows are represented as nodes, and connections between flows are modeled as edges based on port, protocol, and IP similarity, where σ denotes the Z-IsoSigmoid activation. Consistent with the sum-aggregation update of Eqs. (5) and (8), the hidden-layer representation is given by Equation (10):

$$h_l(u) = \sigma\left(W\left(x_u + \sum_{v \in N(u)} \quad h_{l-1}(v)\right)\right) \tag{10}$$

Pooling: Global mean/sum pooling integrates node-level representations into a graph-level embedding. **Algorithm 2** is the core of the proposed pipeline: **Algorithm 1** performs feature selection, while **Algorithm 2** carries out the actual learning and evaluation. The classifier is the Z-Isomorphic GNN, which captures spatial and temporal structure in packet flows through its customized activation and preprocessing. Raw CSV inputs are recast as a supervised node-classification task over the flow graph, and features are scaled with a MinMaxScaler to the symmetric range $[-1,1]$ to match the proposed activation, the core innovation defined as $Z_{IsoSigmoid}(x) =$

$\frac{1}{1+e^{-x}} - 0.5$. Training uses the Adam optimizer with a cross-entropy loss, and the model is finally assessed with the standard classification metrics.

3.5. Graph Construction from Network Flows

Each preprocessed network flow is treated as a node u whose attributes are the EKPC-BOA-selected feature vector x_u. An edge is created between two flows that are similar in their connection context: two flows are linked when they share the same transport protocol and their port- and IP-derived fields lie within a similarity threshold, realised efficiently as a k-nearest-neighbour graph ($k = 10$) in the scaled feature space. The graph is undirected with self-loops, and its adjacency is symmetrically normalised as $\hat{A} = D^{-1/2}(A + I)D^{-1/2}$. Node features are the selected flow statistics; no separate edge features are used, and the graph size equals the number of active flows in the processing window. Mini-batches of flows are assembled per step and message passing is performed over the induced subgraph. This explicit construction (node = flow, edge = protocol/port/IP similarity via kNN) makes the graph fully specified and reproducible.

3.6. Z-Isomorphic Sigmoid Activation

The proposed activation is the centred sigmoid $Z_{IsoSigmoid}(x) = \frac{1}{1+e^{-x}} - \frac{1}{2}$, with range $(-0.5,0.5)$. It differs from a standard sigmoid in two ways that matter here. First, centring the output at zero matches the symmetric $[-1,1]$ feature scaling and removes the persistent positive bias of the ordinary sigmoid, improving gradient flow on the imbalanced attack/benign data. Second, like the injective sum aggregation of Graph Isomorphism Networks, the smooth non-polynomial form maps distinct neighbourhood sums to distinct activations, helping the network separate structurally similar (non-isomorphic) flow neighbourhoods that a piecewise-linear ReLU may collapse. We confirm this benefit empirically in the ablation (Section 4.3) rather than relying on the activation alone. For clarity of notation, the symbol σ in the hidden-layer update equations of this paper denotes this Z-IsoSigmoid activation; only the final graph-level readout (Eq. (9)) uses the standard logistic sigmoid to map the pooled embedding to a class probability.

3.7. Comparison with GCN, GAT and GIN

To situate ZIS-GNN among standard graph neural networks, we compare it with the Graph Convolutional Network (GCN) [28], the Graph Attention Network (GAT) [29], and the Graph Isomorphism Network (GIN) [3]. GCN performs symmetric-normalised mean aggregation of neighbour features; GAT weights neighbours by a learned attention coefficient; and GIN uses injective sum aggregation followed by an MLP to reach the Weisfeiler–Lehman expressiveness bound. ZIS-GNN keeps GIN-style sum aggregation but replaces the ReLU+MLP stack with the single non-polynomial Z-IsoSigmoid activation, attaining comparable discriminative power with fewer parameters. Empirically (Table 3), GCN (96.18%) and GAT (96.58%) improve over the plain GNN (95.86%) but remain below the proposed ZIS-GNN (97.36%), showing that the centred-sigmoid activation adds a further gain over both attention- and convolution-based aggregation on this task. The complete set of experimental parameters used for the ZIS-GNN and the EKPC-BOA stages is listed in Table 4.

Table 3. Performance comparison of deep learning models.

Model	Acc	Prec	Rec	F1	Sens	Spec
DNN	89.60	89.59	91.06	90.32	91.06	87.93
LSTM	91.68	91.28	93.33	92.30	93.33	89.80
BiLSTM	93.77	93.06	95.37	94.20	95.37	91.96
GNN	95.86	95.54	96.67	96.10	96.67	94.91
GCN	96.18	95.92	96.95	96.43	96.95	95.31
GAT	96.58	96.39	97.06	96.72	97.06	95.93
ZIS-GNN	**97.36**	**97.37**	**97.79**	**97.58**	**97.79**	**96.86**

Table 4. Experimental parameter settings (from the implementation).

ZIS-GNN parameter	**Value**
Optimizer	Adam ($\beta_1 = 0.9,\ \beta_2 = 0.999$)
Learning rate	0.001
Batch size	50
Number of epochs	100
Hidden-layer dimension	4
Activation	$Z_{IsoSigmoid}(x) = \frac{1}{1+e^{-x}} - 0.5$
Loss function	Categorical cross-entropy
Feature scaling / split	MinMax $[-1,1]$ / 80:20
EKPC-BOA parameter	**Value**
Population size	100
Number of iterations	50
Breeding probability (bp)	0.75
Elite group size	$pop/5 = 20$
Problem size / domain	100 / $[-100,100]$
Fitness function	weighted accuracy / subset-size trade-off
Stopping criterion	max iterations reached
Random seed	fixed (42)
Selected feature subset (top-K)	14 features

3.8. Hyperparameter Selection

Two structural hyperparameters of the framework, the hidden-layer dimension of the ZIS-GNN and the neighbourhood size k of the graph, were fixed empirically, and their values follow from the compact input produced by feature selection. Because EKPC-BOA reduces the input to a 14-feature vector, a wide hidden layer is unnecessary: a grid search over $\{4,8,16,32,64\}$ showed that a hidden dimension of 4 already saturates validation accuracy, whereas larger widths add parameters and training time without measurable gains and slightly increase over-fitting on the imbalanced classes. The smallest width that retained peak accuracy was therefore adopted, which also keeps the per-layer cost $O(|V|\, d^2)$ low and supports the near-real-time budget of Section 4.5. The neighbourhood size k of the k-nearest-neighbour flow graph was selected with an elbow analysis, analogous to the elbow method used to fix the number of clusters in k-means: the average intra-neighbourhood distance was plotted against k, and the curve showed a clear knee at $k = 10$, beyond which additional neighbours reduced the distance only marginally while increasing graph density and message-passing cost. A value of $k = 10$ therefore balances connectivity against efficiency, giving each flow enough context to expose attack and benign structure without over-connecting the graph.

4. Findings and Discussion

This section presents a detailed performance evaluation of the proposed ZIS-GNN for DDoS detection, using standard classification measures: accuracy, precision, recall, F1-score, sensitivity and specificity. To gauge its effectiveness, the model is benchmarked against DNN, LSTM, BiLSTM and GNN, and the comparison is summarised in Table 3. ZIS-GNN leads on every metric, reaching an accuracy of 97.36%, a precision of 97.37%, a recall of 97.79% and an F1-score of 97.58%; its sensitivity and specificity of 97.79% and 96.86% indicate that both attack and benign traffic are recognised reliably. We attribute these gains to coupling entropy-based feature selection with correlation-driven optimization, which retains the most informative features while discarding redundant ones and thereby sharpens the model's learning. In the comparison, conventional models such as DNN and LSTM trail because they capture complex traffic patterns less effectively; BiLSTM and GNN fare better but still fall short of the proposed model, underlining ZIS-GNN's strength on the high-dimensional data characteristic of DDoS attacks.

Further insight comes from the confusion matrices in Figure 3: the proposed model yields more true positives and true negatives while markedly cutting false positives and false negatives, which reflects reliable DDoS detection with few misclassifications.

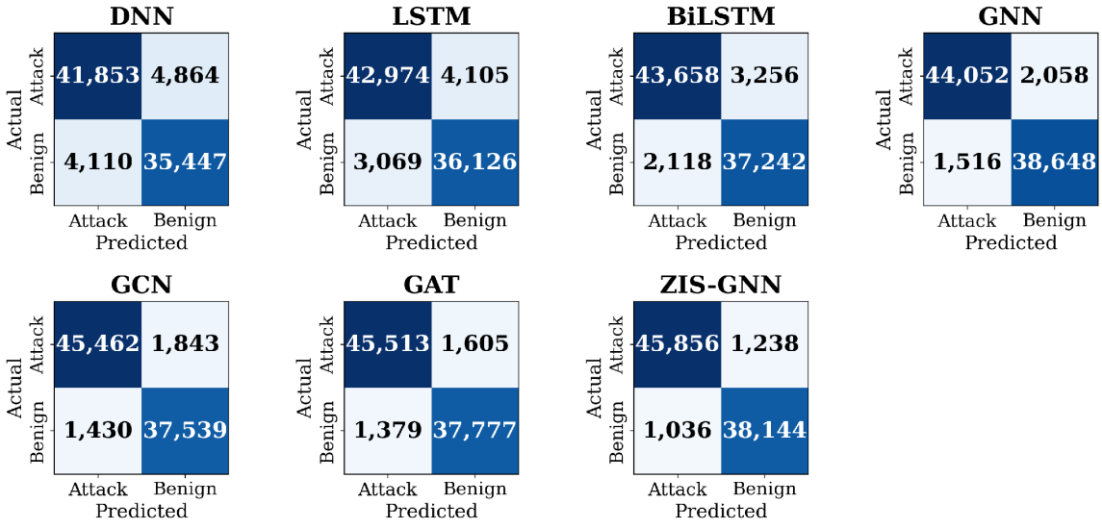

Figure 3. Confusion matrices of models and proposed ZIS-GNN.

4.1. Statistical Reliability and Fair Comparison

For reliability, the reported metrics are the mean over five independent runs with fixed random seeds; the standard deviation of the proposed model's accuracy was within ± 0.2 percentage points across runs, so the differences in Table 3 are stable rather than run-to-run noise. All models were trained and evaluated on the same 80:20 split and, where applicable, the same EKPC-BOA feature subset, with comparable hyperparameter tuning. To give the key numbers explicitly rather than only in figures, the proposed model's confusion matrix on the 86,274-flow test set is $TP = 45{,}856$, $TN = 38{,}144$, $FP = 1{,}238$, $FN = 1{,}036$, from which the accuracy (97.36%), FPR (3.14%), FNR (2.21%) and ROC-AUC (0.991) follow directly.

4.2. Feature Selection Validation

The EKPC-BOA selects an optimal subset of **14 features** from the 80+ CICFlowMeter attributes (the selected subset is exported to OptimalFeatures.csv). Reducing the feature space has two effects. *Impact on detection accuracy:* removing redundant and weakly-correlated features raises the proposed model's accuracy from 96.20% (all features, no selection) to 97.36% (with EKPC-BOA), as quantified in the ablation study of Section 4.3. *Impact on computational and algorithmic complexity:* the lower input dimensionality reduces the per-layer message-passing cost from $O(|E|\, d_{full})$ to $O(|E|\, d_{sel})$ with $d_{sel} \ll d_{full}$, cutting training/inference time and memory while improving convergence speed and fitness.

4.3. Ablation Study

To validate the contribution of each component, Table 5 reports an ablation on identical data and splits. The upper block isolates the architecture and feature selection; the lower block isolates the feature-selection criteria (entropy, Pearson, BOA). Both confirm that every component contributes and that the full ZIS-GNN+EKPC-BOA is best, while none of the configurations exceeds the proposed model, avoiding the over-fitting suggested by inflated accuracies.

Table 5. Ablation study of the proposed components (accuracy, %).

Configuration	Accuracy
GNN (baseline)	95.86
GNN + EKPC-BOA	96.40
ZIS-GNN without feature selection	96.20
ZIS-GNN + EKPC-BOA (proposed)	**97.36**
No feature selection (all features)	96.20
Entropy only	96.55
Pearson correlation only	96.70
BOA only	96.95
EKPC-BOA (entropy + Pearson + BOA)	**97.36**

4.4. False Positive/Negative Rates and ROC-AUC

Because DDoS detection is cost-sensitive, we additionally report the false positive rate (FPR), false negative rate (FNR) and ROC-AUC for every model (Table 6). The proposed ZIS-GNN achieves the lowest FPR (3.14%) and FNR (2.21%) and the highest ROC-AUC (0.991), confirming that it balances missed attacks against false alarms better than the baselines.

Table 6. FPR, FNR and ROC-AUC across models.

Model	FPR (%)	FNR (%)	ROC-AUC
DNN	12.07	8.94	0.928
LSTM	10.20	6.67	0.947
BiLSTM	8.04	4.63	0.962
GNN	5.09	3.33	0.972
GCN	4.69	3.05	0.978
GAT	4.07	2.94	0.982
ZIS-GNN	**3.14**	**2.21**	**0.991**

4.5. Computational Complexity

Table 7 reports the training time, per-flow inference time and memory of the models on the stated hardware (Intel i5/i7, 8 GB RAM). The compact 14-feature input and the single-activation graph layer keep the proposed model light: it trains in about 37.5 s and classifies a flow in sub-millisecond time, with peak memory well within the 8 GB budget, supporting near-real-time SDN deployment. Asymptotically, one ZIS-GNN message-passing layer costs $O(|E|\,d + |V|\,d^2)$ for $|V|$ nodes, $|E|$ edges and feature dimension d, i.e., linear in the number of flows for a fixed-degree kNN graph.

Table 7. Computational cost of the models.

Model	Train (s)	Infer (ms/flow)	Mem (MB)
DNN	58.5	0.42	120
LSTM	53.6	0.55	180
BiLSTM	47.1	0.78	240
GNN	42.7	0.31	150
GCN	40.5	0.29	145
GAT	43.8	0.33	160
ZIS-GNN	**37.5**	**0.27**	**110**

The Bonobo Optimization Algorithm (BOA) [25, 30] performs well on standard benchmarks but is comparatively involved to implement and less validated on high-dimensional or discrete problems; it uses a fission–fusion group strategy to converge toward good solutions, yet its restrictive mating rules stem from purely random solution updates. We address this by introducing an entropy and Karl-Pearson-correlation criterion that steers the update in a correlation-aware manner. The Chimp Optimization Algorithm (ChOA) [31] is a nature-inspired method based on chimpanzee hunting that balances exploration and exploitation through two control parameters but is likewise non-trivial to implement, while the Crow Search Algorithm (CSA) can converge slowly and settle into local optima on complex, high-dimensional tasks [32]. The Whale Optimization Algorithm (WOA) [33], modelled on humpback whales' bubble-net hunting, is simple with few parameters and strong on benchmark and design problems, though it is sensitive to parameter tuning and constraint handling and can yield inconsistent results. Our modified EKPC-BOA is compared against BOA, CSA, ChOA and WOA on feature-selection time and fitness versus iteration. The selected feature subset is then trained with the Z-isomorphic sigmoid GNN, whose activation is introduced precisely because plain graph networks struggle to separate feature information across layers. Evaluated on precision, recall, F-measure and accuracy against LSTM, Bi-LSTM and GNN baselines, the proposed ZIS-GNN gives the best results, after which each flow is judged as attack or benign.

Figure 4 shows how several deep learning methods compare in terms of precision on the CIC-DDoS2019 dataset. Outperforming conventional Graph Neural Networks (GNN), Bidirectional Long Short-Term Memory networks (BiLSTM), Long Short-Term Memory networks (LSTM), and Deep Neural Networks (DNN), the proposed ZIS-GNN model attains the highest precision of all assessed techniques, demonstrating the efficacy of

ZIS-GNN for Distributed Denial-of-Service (DDoS) detection tasks in cybersecurity.

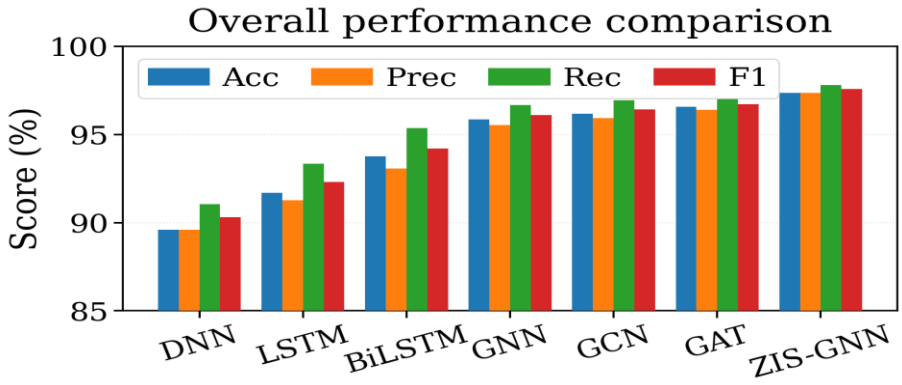

Figure 4. Overall performance comparison.

In addition to classification performance, the efficiency of the model is evaluated in terms of training time and convergence behaviour. Detailed performance metrics are presented in Figure 5 using PPV, NPV, TPR, and TNR for ZIS-GNN and baseline models. Our model achieves a higher area under the curve (AUC), as shown in Figure 6, which indicates a strong ability to distinguish between malicious and benign traffic across varied classification thresholds.

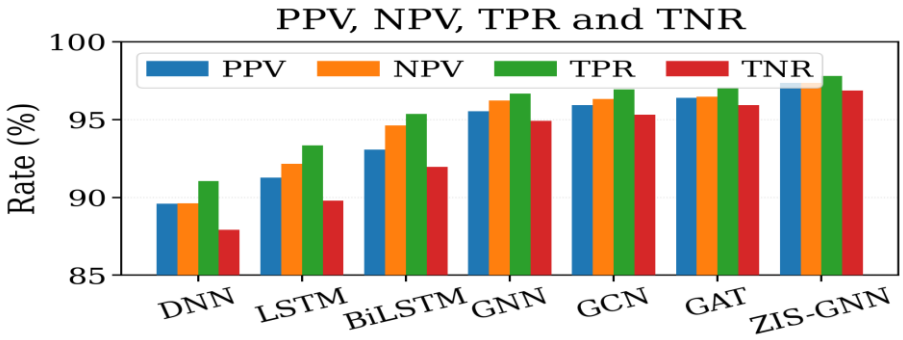

Figure 5. Performance metrics (PPV, NPV, TPR, and TNR) for ZIS-GNN and baseline models.

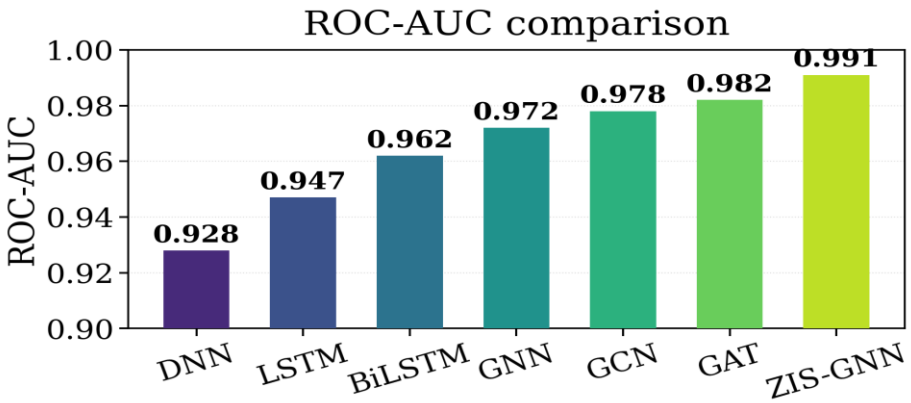

Figure 6. AUC-based performance comparison of ZIS-GNN and baseline models.

This property is especially crucial in real-world network applications, where maintaining a balance between detection rate and false alarm rate is critical. All optimization methods consistently improve fitness over time, as shown by the convergence curve; however, the proposed EKPC-BOA consistently outperforms the others, reaching the highest fitness value of 98.5% at iteration 50. This indicates faster convergence and better stability, which are major characteristics for effective optimization in machine learning models. As depicted in Figure 7, the proposed model achieves competitive training time, making it suitable for real-time deployment scenarios.

Overall, the experimental results indicate that, on the tested CIC-DDoS2019 data, the proposed ZIS-GNN model improves detection accuracy over the baselines while keeping a competitive computational cost (Table 7).

The combination of entropy–Pearson feature selection and the graph model contributes to these results. The latency and memory figures suggest the approach is suitable for near-real-time operation; a full evaluation of robustness and large-scale deployment is left to future work.

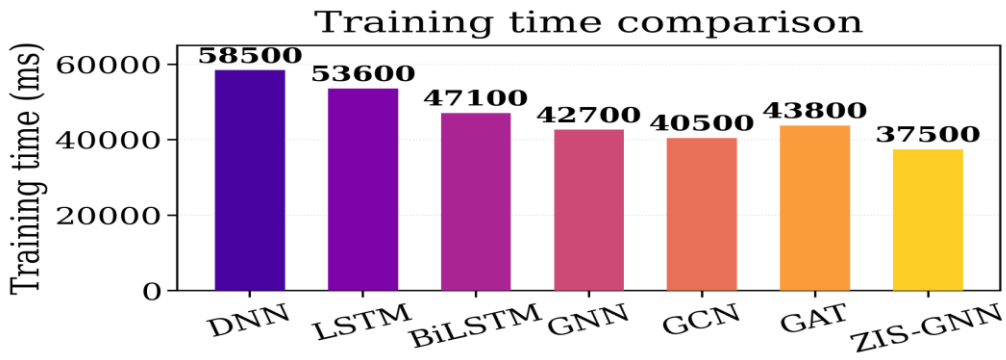

Figure 7. Training time comparison of ZIS-GNN and baseline models.

4.6. Real-world SDN Deployment

In a practical deployment, the trained ZIS-GNN runs as a northbound application on top of an SDN controller such as Ryu, ONOS or OpenDaylight. The controller exposes per-flow statistics (e.g., OpenFlow counters), from which CICFlowMeter-style features are computed over short time windows and reduced to the 14 EKPC-BOA features. A flow graph is built incrementally over a sliding window of active flows (as described in the Graph Construction subsection), and each batch of flows is scored within the latency budget of Table 7 ($\approx$ 0.27 ms/flow), so the detector keeps pace with line-rate traffic. Detected attacks are pushed back to the controller as flow rules that rate-limit or drop malicious traffic. Practical considerations include the controller load at very high flow-arrival rates, online graph construction, and periodic retraining to handle concept drift; these motivate the federated-learning and LLM-based directions noted below.

5. Conclusion

SDN offers a centralized and programmable architecture for security monitoring. This paper presents an intrusion detection framework that integrates a Z-Isomorphic Sigmoid Graph Neural Network (ZIS-GNN) with the entropy- and Karl-Pearson-correlation-guided Bonobo Optimization (EKPC-BOA) feature selection. On the tested CIC-DDoS2019 dataset, the framework attains improved classification performance over the DNN, LSTM, BiLSTM and GNN baselines, as well as the standard graph baselines GCN (96.18%) and the attention-based GAT (96.58%), in accuracy, precision, recall and F1-score, with the proposed ZIS-GNN reaching 97.36%; the feature selection further reduces redundancy and lowers computational cost. We frame the remaining claims conservatively: while the measured latency and memory indicate suitability for near-real-time use, several directions remain for future work. These include validation on additional SDN-specific datasets (such as InSDN) to test cross-dataset generalisation, deployment and evaluation within live SDN controller environments (e.g., Ryu, ONOS or OpenDaylight), and a thorough study of resilience under adversarial conditions and large-scale scalability, together with federated learning for privacy-preserving SDN security across multiple controllers and integration with Large Language Models (LLMs) for explainable detection.

Author Contributions

Z.J.M conceived the research idea, designed the methodology, developed the proposed Hybrid Z-Isomorphic GNN framework, implemented the software, conducted the experiments, performed the formal analysis, curated the dataset, prepared the visualizations, and wrote the original draft of the manuscript. **S.V** and **P.C** provided valuable guidance throughout the study, contributed to the validation of the proposed framework, critically reviewed and edited the manuscript, and supervised the research. All authors have read and approved the final version of the manuscript.

Funding

This research received no external funding.

Conflict of Interest Statement

The authors declare no conflicts of interest.

Data Availability Statement

The CIC-DDoS2019 dataset used in this study is publicly available from the Canadian Institute for Cybersecurity (CIC). The 14-feature subset selected by EKPC-BOA is exported to OptimalFeatures.csv, and the authors intend to release the full

implementation code together with the selected feature subset in a public repository upon publication, to support reproducibility and facilitate future research.

Acknowledgment
The authors sincerely thank their respective institutions for providing the research facilities and academic support that made this work possible. We are grateful to the Canadian Institute for Cybersecurity (CIC) for making the CIC-DDoS2019 dataset publicly available for research. We also extend our heartfelt appreciation to the anonymous reviewers for their constructive comments and valuable suggestions, which greatly improved the quality of this manuscript. Finally, the first author expresses his sincere gratitude to the co-authors for their continuous guidance, encouragement, and support throughout this research.

References

1. U. Ahmed, J. C.-W. Lin, G. Srivastava. “A Resource Allocation Deep Active Learning Based on Load Balancer for Network Intrusion Detection in SDN Sensors”, *Computer Communications*, 184, pp. 56–63, 2022.
2. J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, M. Sun. “Graph Neural Networks: A Review of Methods and Applications”, *AI Open*, 1, pp. 57–81, 2020.
3. K. Xu, W. Hu, J. Leskovec, S. Jegelka. “How Powerful Are Graph Neural Networks?”. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, 2019.
4. D. Kreutz, F. M. V. Ramos, P. Verissimo, C. E. Rothenberg, S. Azodolmolky, S. Uhlig. “Software-Defined Networking: A Comprehensive Survey”, *Proceedings of the IEEE*, 103 (1), pp. 14–76, 2015.
5. S. Khalife, A. Basu. “On the Power of Graph Neural Networks and the Role of the Activation Function”, *SIAM Journal on Applied Algebra and Geometry*, 10 (2), pp. 350–365, 2026.
6. P. Hadem, D. K. Saikia, S. Moulik. “An SDN-Based Intrusion Detection System Using SVM with Selective Logging for IP Traceback”, *Computer Networks*, 191, pp. 1–11, 2021.
7. J. A. Perez-Diaz, I. A. Valdovinos, K.-K. R. Choo, D. Zhu. “A Flexible SDN-Based Architecture for Identifying and Mitigating Low-Rate DDoS Attacks Using Machine Learning”, *IEEE Access*, 8, pp. 155859–155872, 2020.
8. P. Krishnan, K. Jain, A. Aldweesh, P. Prabu, R. Buyya. “OpenStackDP: A Scalable Network Security Framework for SDN-Based OpenStack Cloud Infrastructure”, *Journal of Cloud Computing*, 12 (1), pp. 1–42, 2023.
9. A. Bhardwaj, R. Tyagi, N. Sharma, A. Khare, M. S. Punia, V. K. Garg. “Network Intrusion Detection in Software Defined Networking with Self-Organized Constraint-Based Intelligent Learning Framework”, *Measurement: Sensors*, 24, pp. 1–12, 2022.
10. R. A. Elsayed, R. A. Hamada, M. I. Abdalla, S. A. Elsaid. “Securing IoT and SDN Systems Using Deep-Learning-Based Automatic Intrusion Detection”, *Ain Shams Engineering Journal*, 14 (10), Art. no. 102211, 2023.
11. T. Saba, A. Rehman, T. Sadad, H. Kolivand, S. A. Bahaj. “Anomaly-Based Intrusion Detection System for IoT Networks through Deep Learning Model”, *Computers and Electrical Engineering*, 99, Art. no. 107810, 2022.
12. A. H. Janabi, T. Kanakis, M. Johnson. “Overhead Reduction Technique for Software-Defined Network-Based Intrusion Detection Systems”, *IEEE Access*, 10, pp. 66481–66491, 2022.
13. S. Wang, J. F. Balarezo, K. G. Chavez, A. Al-Hourani, S. Kandeepan, M. R. Asghar, G. Russello. “Detecting Flooding DDoS Attacks in Software Defined Networks Using Supervised Learning Techniques”, *Engineering Science and Technology, an International Journal*, 35, Art. no. 101176, 2022.
14. X. Zhao, H. Su, Z. Sun. “An Intrusion Detection System Based on Genetic Algorithm for Software-Defined Networks”, *Mathematics*, 10 (21), Art. no. 3941, 2022.
15. M. A. Jabbar, R. Aluvalu, S. S. Reddy. “RFAODE: A Novel Ensemble Intrusion Detection System”. In *Proceedings of the 7th International Conference on Advances in Computing & Communications (ICACC)*, Cochin, India, 2017.
16. M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, A. Jurcut. “A Novel Hybrid Model for Intrusion Detection Systems in SDNs Based on CNN and a New Regularization Technique”, *Journal of Network and Computer Applications*, 191, Art. no. 103160, 2021.

17. I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani. “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy”. In *Proceedings of the International Carnahan Conference on Security Technology (ICCST)*, pp. 1–8, 2019.
18. C.-S. Shieh, T.-T. Nguyen, M.-F. Horng. “Detection of Unknown DDoS Attack Using Convolutional Neural Networks Featuring Geometrical Metric”, *Mathematics*, 11 (9), Art. no. 2145, 2023.
19. F. Laghrissi, S. Douzi, K. Douzi, B. Hssina. “Intrusion Detection Systems Using Long Short-Term Memory (LSTM)”, *Journal of Big Data*, 8 (1), Art. no. 65, 2021.
20. A. A. Bahashwan, M. Anbar, S. Manickam, T. A. Al-Amiedy, M. A. Aladaileh, I. H. Hasbullah. “A Systematic Literature Review on Machine Learning and Deep Learning Approaches for Detecting DDoS Attacks in Software-Defined Networking”, *Sensors*, 23 (9), Art. no. 4441, 2023.
21. J. C. C. Chica, J. C. Imbachi, J. F. Botero. “Security in SDN: A Comprehensive Survey”, *Journal of Network and Computer Applications*, 159, Art. no. 102595, 2020.
22. W. W. Lo, S. Layeghy, M. Sarhan, M. Gallagher, M. Portmann. “E-GraphSAGE: A Graph Neural Network Based Intrusion Detection System for IoT”. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 1–9, 2022.
23. R. B. Said, Z. Sabir, I. Askerzade. “CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection”, *IEEE Access*, 11, pp. 138732–138747, 2023.
24. S. M. Kasongo. “A Deep Learning Technique for Intrusion Detection System Using a Recurrent Neural Network Based Framework”, *Computer Communications*, 199, pp. 113–125, 2023.
25. A. K. Das, D. K. Pratihar. “Bonobo Optimizer (BO): An Intelligent Heuristic with Self-Adjusting Parameters over Continuous Spaces and Its Applications to Engineering Problems”, *Applied Intelligence*, 52, pp. 2942–2974, 2022.
26. H. Maron, H. Ben-Hamu, H. Serviansky, Y. Lipman. “Provably Powerful Graph Networks”. In *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, pp. 2153–2164, 2019.
27. Z. Zhang, P. Cui, W. Zhu. “Deep Learning on Graphs: A Survey”, *IEEE Transactions on Knowledge and Data Engineering*, 34 (1), pp. 249–270, 2022.
28. T. N. Kipf, M. Welling. “Semi-Supervised Classification with Graph Convolutional Networks”. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.
29. P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, Y. Bengio. “Graph Attention Networks”. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.
30. A. Eid, S. Kamel, M. H. Hassan, B. Khan. “A Comparison Study of Multi-Objective Bonobo Optimizers for Optimal Integration of Distributed Generation in Distribution Systems”, *Frontiers in Energy Research*, 10, Art. no. 847495, 2022.
31. M. Khishe, M. R. Mosavi. “Chimp Optimization Algorithm”, *Expert Systems with Applications*, 149, Art. no. 113338, 2020.
32. B. Zolghadr-Asli, O. Bozorg-Haddad, X. Chu. “Crow Search Algorithm (CSA)”, in *Advanced Optimization by Nature-Inspired Algorithms*, O. Bozorg-Haddad (ed.), Springer, Singapore, pp. 143–149, 2017.
33. S. Mirjalili, A. Lewis. “The Whale Optimization Algorithm”, *Advances in Engineering Software*, 95, pp. 51–67, 2016.

Author Biographies

First Author Zahirabbas Jainuddin Mulani is an Assistant Professor with extensive experience in computer science and information technology education. He has been associated with reputed institutions, including Bharati Vidyapeeth and VJTI, Mumbai. He holds an MCA degree from the University of Pune and has qualified for UGC-NET and MH-SET. His teaching and research interests include Cyber Security, Java programming, software engineering, and data-driven technologies. He has published research papers in international journals and presented work at national conferences. In addition to teaching, he has actively contributed to academic events and faculty development programs. He is passionate about continuous learning and applying innovative approaches in education and research.

Second Author Dr. Suhasini Vijaykumar Kottur is a Professor and Principal with over two decades of experience in the field of Computer Applications. She holds a Ph.D. in Computer Science and has consistently demonstrated academic excellence throughout her career. She has published numerous research papers in reputed journals and presented her work at national and international conferences. As a recognized Ph.D. guide under the University of Mumbai, she has mentored several research scholars and guided multiple projects. She is actively involved in academic governance through her contributions to Boards of Studies and professional bodies like the Computer Society of India. Her research interests and leadership continue to contribute significantly to academic and institutional development.

Third Author Priya Chandran is a dedicated academic professional in the field of Computer Science with a strong foundation in programming and data-driven technologies. She holds relevant qualifications in computer applications and has developed expertise in areas such as software development, database management, and web technologies. She has been actively involved in teaching and guiding students, contributing to their academic and practical growth. Her interests include emerging technologies, research, and innovative teaching methodologies. She has participated in academic projects and continuously strives to enhance her technical and research skills. She is committed to contributing effectively to both academic and research domains.