Innovative: International Multi-disciplinary Journal of Applied Technology
(ISSN 2995-486X) VOLUME 03 ISSUE 09, 2025

Genetic Algorithm–Based Hyperparameter Optimization for 1D-CNN NIDS

[+] ***Juraev G.U.*[1], Kilichev D.[2], Mukhamadiyev A.N.[3], Saydirasulov S.N.[3], Turimov D.M.[3]**
[1] National University of Uzbekistan named after Mirzo Ulugbek, Uzbekistan
[2] Samarkand Branch of Tashkent University of Economics, Uzbekistan
[3] Department of IT Convergence Engineering Gachon University, Gyeongi-do, South Korea
[+] dusmurod@gacho.ac.kr

Abstract:

Abstract. This paper investigates Genetic Algorithm (GA)–driven hyperparameter optimization for one-dimensional Convolutional Neural Networks (1D-CNNs) in network intrusion detection systems (NIDS). Building on prior evidence that evolutionary search markedly improves deep models for NIDS, we optimize nine key hyperparameters (filters, kernel size, pooling size, dense depth/width, dropout, learning rate, batch size, and epochs) and train/evaluate on UNSW-NB15, CIC-IDS2017, and NSL-KDD using standard metrics: accuracy, loss, precision, recall, and F1-score. Our GA framework encodes hyperparameters as chromosomes and evolves candidates via selection, crossover, and mutation to maximize validation performance. Experiments show that GA-tuned 1D-CNNs consistently outperform non-optimized baselines across datasets; on UNSW-NB15, for example, GA attains ~99.31% accuracy, aligning with the best reported performance for GA-optimized 1D-CNNs. Results highlight that GA delivers robust gains with practical compute budgets while preserving strong precision-recall trade-offs, and that effectiveness can vary by dataset characteristics. Overall, GA-based hyperparameter optimization offers a simple, reproducible path to higher accuracy and reliability in deep learning–based NIDS, advancing the development of adaptable intrusion-detection solutions for evolving cyber threats.

Keywords: intrusion-detection system (IDS); network intrusion detection system (NIDS); convolutional neural network (CNN); one-dimensional convolutional neural network (1D-CNN); hyperparameter optimization; genetic algorithm (GA).

1. Introduction

As networked services proliferate, cyberattacks have grown in volume and sophistication, making the protection of digital assets a central concern for organizations and individuals. Network intrusion detection systems (NIDSs) are deployed to monitor traffic and identify malicious activity in real time, yet traditional rule- and signature-based approaches struggle to keep pace with novel or evolving threats. Deep learning, and in particular one-dimensional convolutional neural networks (1D-CNNs), has shown strong promise for learning discriminative patterns from sequential network-flow features [1]. However, the performance of a 1D-CNN-based NIDS depends critically on hyperparameters such as the number of filters, kernel and pooling sizes, depth and width of dense layers, dropout rate, learning rate, batch size, and training epochs. Manual tuning of these choices is labor-intensive, dataset-dependent, and prone to suboptimal configurations. To address this challenge, this paper focuses on the Genetic Algorithm (GA) as an evolutionary search method for hyperparameter optimization. GA's selection, crossover, and mutation operators provide an adaptive balance between exploration and exploitation, offering a practical route to discover performant configurations for 1D-CNN intrusion detectors while reducing human effort and trial-and-error [2].

The aim of this study is to develop and evaluate a GA-driven framework that automatically optimizes the hyperparameters of a 1D-CNN for network intrusion detection across widely used benchmarks [3]. We investigate whether GA can consistently enhance accuracy and related detection metrics by navigating the high-dimensional configuration space more efficiently than manual or naive search. The study contributes an end-to-end, reproducible optimization pipeline that encodes nine key hyperparameters and evolves candidate solutions to maximize validation performance; a comprehensive empirical assessment on UNSW-NB15, CIC-IDS2017, and NSL-KDD demonstrating improvements over non-optimized baselines; and an analysis of GA behavior that clarifies practical settings for population size, crossover and mutation rates, and early stopping to balance compute cost with accuracy. Beyond raw performance gains, the framework is designed to be adaptable to other deep architectures and operational contexts, reducing the time and expertise required to deploy reliable, up-to-date NIDS models in practice [4].

The remainder of the paper proceeds as follows. Section 2 reviews related work on deep learning–based intrusion detection and hyperparameter optimization methods, positioning our approach in the literature. Section 3 details the proposed GA optimization framework, including the encoding of hyperparameters and the selection, crossover, mutation, and elitism strategies. Section 4 presents the 1D-CNN architecture, datasets, and preprocessing pipeline used in our experiments. Section 5 describes the experimental setup, evaluation metrics, and results obtained by the GA-optimized models. Section 6 discusses the findings, analyzes GA convergence and sensitivity, and reflects on implications for practical deployment. Section 7 concludes with a summary of contributions and outlines directions for future research.

2. RELATED WORK

Research on hyperparameter optimization for intrusion detection has evolved from manual, heuristic practices to principled search and metaheuristic methods. Early deep learning–based NIDS studies often relied on manual tuning or coarse grid/random search, approaches that are labor-intensive and prone to suboptimal configurations because the search space is large and dataset-specific [5]. The foundational problem is that 1D-CNN performance hinges on choices such as the number of filters, kernel and pooling sizes, dense-layer depth and width, dropout, learning rate, batch size, and epochs; these are not learned from data but fixed before training, so poor settings can hurt convergence and generalization. Prior work therefore framed hyperparameter selection as an optimization problem to be solved with data-driven search rather than intuition alone, motivating the move toward more efficient optimizers for CNN-based NIDS [6].

Randomized search improved coverage of high-dimensional spaces under fixed budgets by sampling configurations and selecting those that validated well, while successive-halving and bandit-inspired methods such as Hyperband allocated budget adaptively to promising candidates [7]. Bayesian optimization, including Tree-Structured Parzen Estimators, brought a model-based view that balances exploration and exploitation and handles conditional hyperparameters. Alongside these, neuroevolution and architecture search explored topology and weight spaces jointly to trade detection performance against model complexity [8]. Collectively, these strands established that principled search reliably outperforms manual tuning for NIDS hyperparameters and that the best method can be context dependent, varying with dataset characteristics and model capacity [9].

Within metaheuristics, Genetic Algorithms (GA) and Particle Swarm Optimization (PSO) have been especially influential due to their ability to navigate mixed discrete–continuous spaces and to maintain diversity while seeking high-fitness regions [10]. GA encodes hyperparameters as chromosomes and iteratively applies selection, crossover, and mutation, offering strong exploratory behavior that helps avoid premature convergence [11]. PSO represents configurations as particles updated toward personal and global bests, often yielding faster convergence on smooth fitness landscapes. In the specific context of 1D-CNN-based NIDS, both GA and PSO have been shown to produce substantial gains in accuracy, precision, recall, and F1 over non-optimized baselines across standard benchmarks such as UNSW-NB15, CIC-IDS2017, and NSL-KDD, underscoring the value of evolutionary optimization for this task [12].

The present paper builds on this trajectory but narrows the focus to a GA-centric optimization framework tailored to 1D-CNN hyperparameters for NIDS [13]. By concentrating on GA's evolutionary operators and their practical settings, and by evaluating across multiple benchmarks with standardized metrics (accuracy, loss, precision, recall, and F1), the study aims to clarify when and why GA is particularly effective and how its search dynamics can be tuned for robust performance with reasonable compute budgets [14]. In doing so, it complements comparative studies that include PSO by providing a detailed, single-method treatment of GA for IDS hyperparameter optimization while preserving comparability through the same datasets and evaluation criteria [15].

3. OPTIMIZATION OF 1D-CNN HYPERPARAMETERS

A. Hyperparameters of 1D-CNNs

Hyperparameter tuning, often referred to as hyperparameter optimization, is one of the most critical yet challenging stages in designing and training one-dimensional convolutional neural networks (1D-CNNs). Hyperparameters define the structure, learning behavior, and regularization of the model, directly influencing its detection accuracy, convergence rate, and generalization capability [16]. They govern key aspects such as how features are extracted, how fast learning occurs, and how well the model avoids overfitting.

Unlike model parameters, which are learned during training, hyperparameters must be set manually before training begins. These include structural hyperparameters such as the number of convolutional filters, kernel size, pooling size, number of dense layers, and number of neurons; training-related hyperparameters like learning rate, batch size, and number of epochs; and regularization hyperparameters such as dropout rate. Each of these settings interacts with others in nonlinear ways—changing one can significantly affect the performance of the entire model [17].

Finding optimal hyperparameter combinations is therefore a complex search problem. Manual or trial-and-error tuning can be time-consuming and may fail to identify configurations that generalize well across datasets. Moreover, optimal settings vary depending on the dataset's characteristics, such as size, feature distribution, and class imbalance. For these reasons, automated and intelligent optimization approaches are needed to improve model performance and efficiency in intrusion detection [18].

B. Optimization Method: Genetic Algorithm (GA)

The Genetic Algorithm (GA) is an evolutionary computation technique inspired by the process of natural selection and genetic evolution. It is designed to efficiently search large and complex spaces by simulating biological processes such as reproduction, crossover, and mutation. In the context of 1D-CNN hyperparameter optimization, GA is particularly effective because it can explore both discrete and continuous parameter spaces while maintaining diversity among candidate solutions [19].

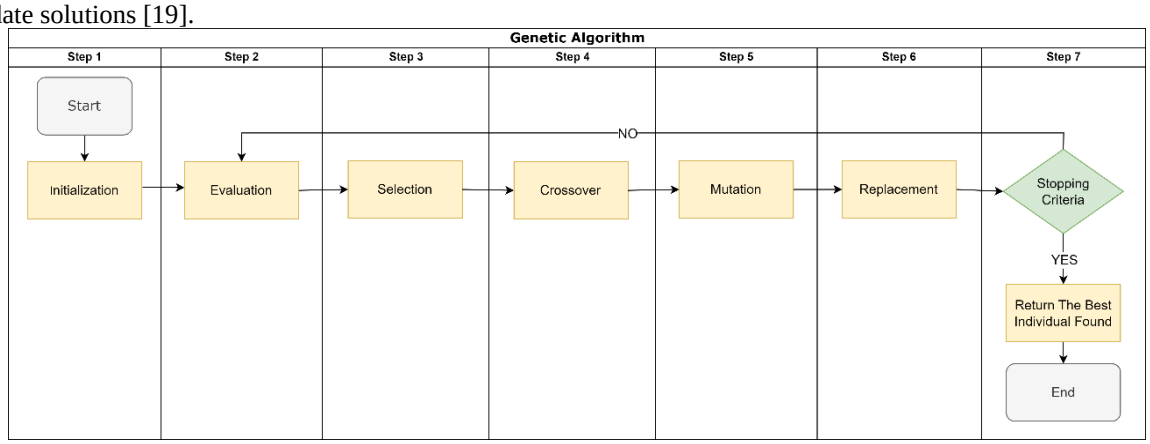


Figure 1. Operational procedure of GA

In a GA-based optimization framework, each candidate solution—known as an individual—is represented as a chromosome that encodes a specific set of hyperparameters. The algorithm begins with an initial population of randomly generated individuals. During each generation, the following steps are performed:

1. Initialization: A population of hyperparameter configurations is generated randomly within predefined bounds.
2. Fitness Evaluation: Each individual's performance is evaluated by training a 1D-CNN with its corresponding hyperparameters and measuring a fitness score, typically based on validation accuracy or F1-score.
3. Selection: The best-performing individuals are selected as parents based on their fitness scores. Selection strategies such as tournament or roulette-wheel selection are often used to ensure that individuals with higher fitness have a greater chance of contributing to the next generation.
4. Crossover: New offspring are produced by combining parts of parent chromosomes. This step allows the algorithm to exploit promising regions of the search space.
5. Mutation: Random changes are introduced into some offspring to maintain diversity and avoid premature convergence.

6. Replacement: The least fit individuals in the population are replaced by the new offspring, ensuring that the population evolves toward better-performing solutions.
7. Termination: The process continues until a stopping criterion is reached, such as a maximum number of generations or a convergence threshold.

The GA optimization process strikes a balance between exploration and exploitation, allowing the search to cover a wide range of possible hyperparameter values while gradually refining the best-performing configurations. Figure 1 illustrates the overall workflow of GA-based hyperparameter optimization for 1D-CNN models, where the evolutionary process iteratively refines the population until an optimal or near-optimal configuration is achieved [20].

By applying GA to tune critical hyperparameters—including filter size, kernel size, number of layers, learning rate, dropout rate, and batch size—the 1D-CNN achieves enhanced detection performance and improved generalization across network intrusion datasets. The adaptive and flexible nature of GA makes it a powerful approach for building optimized and reliable intrusion-detection systems capable of handling diverse and evolving cybersecurity challenges [21].

4. PROPOSED 1D-CNN-BASED NETWORK INTRUSION-DETECTION MODEL

A. Overview of the 1D-CNN architecture

The proposed 1D-CNN receives a one-dimensional sequence of preprocessed features and maps it to a binary decision indicating benign or attack traffic. The input layer expects data shaped as $(sequence\_length,\ 1)$, reflecting a single feature channel over time. Feature extraction is performed by two successive 1D convolutional layers. Each layer applies a bank of learnable filters that slide along the sequence to compute local dot-products and emit feature maps; the number of filters and the kernel size are treated as tunable hyperparameters so the model can adjust both representational capacity and receptive-field length to the statistics of the dataset. After each convolution, a max-pooling layer downsamples the temporal dimension by retaining the maximum within non-overlapping windows. Pooling size is also optimized because it trades spatial resolution for invariance and affects the depth-wise signal-to-noise characteristics learned downstream.

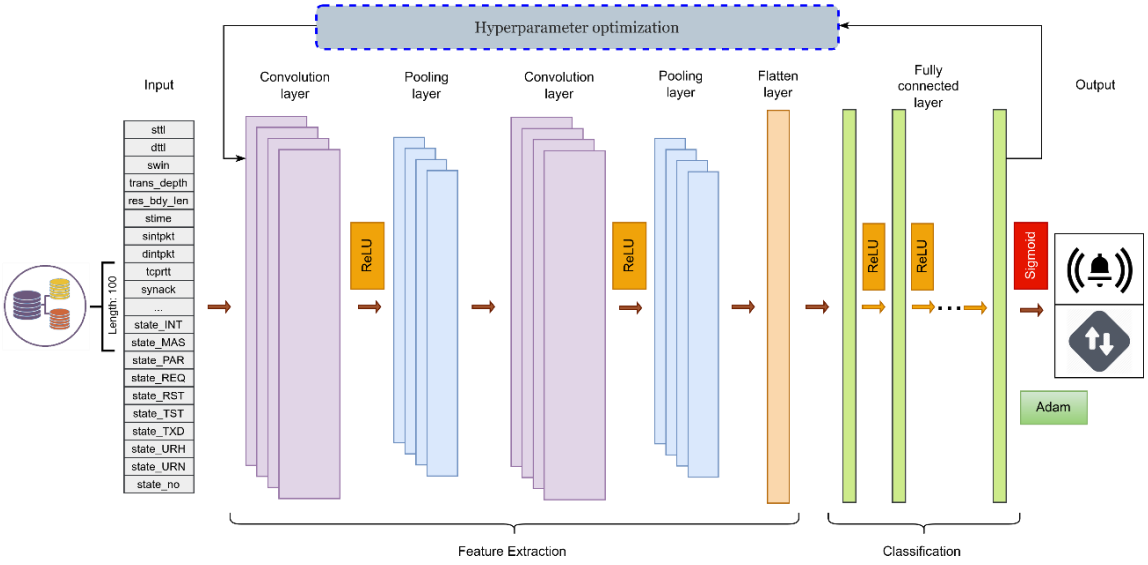


Figure 2. 1D-CNN Model for NIDSs

The convolutional stack is followed by a flattening operation that reshapes the tensor of feature maps into a vector suitable for fully connected processing. One or more dense layers then perform high-level discrimination on the extracted features; both the number of dense layers and the number of neurons per dense layer are optimized to balance expressivity against overfitting. Each dense layer uses ReLU activations to introduce nonlinearity, while dropout is interleaved after dense layers to regularize training by randomly zeroing a fraction of activations, with the dropout rate itself optimized as part of the search. The final output layer employs a sigmoid activation to produce a probability for the positive class in the binary setting. In addition to structural choices, the learning rate, batch size, and number of training epochs are key training-procedure hyperparameters that strongly influence convergence behavior and generalization; these are included in the optimization so the model can adapt end-to-end to each dataset's characteristics.

B. Hyperparameter-optimization problem formulation

We cast hyperparameter selection as a constrained optimization problem. Let $x = [x_1, \ldots, x_9]$ denote the vector of hyperparameters comprising number of convolutional filters (x_1), kernel size (x_2), pooling size (x_3), number of dense layers (x_4), number of neurons per dense layer (x_5), dropout rate (x_6), learning rate (x_7), batch size (x_8), and number of epochs (x_9). The objective is to maximize a fitness function $\phi(x)$ that measures validation performance (e.g., accuracy or F1-score when classes are imbalanced) under dataset-specific preprocessing and a fixed training/validation split. Formally,

$$x^{\backslash *} = arg \max_{x} \phi(x)$$

subject to the discrete and continuous bounds used in prior work on 1D-CNN-based IDS:

$$x_1 \in \{16, 32, 64, 128, 256\},$$
$$x_2 \in \{3, 5, 7, 9, 11\},$$
$$2 < x_3 < 6,$$
$$1 < x_4 < 5,$$
$$x_5 \in \{16, 32, 64, 128, 256\},$$
$$0.1 < x_6 < 0.5,$$
$$10^{-5} < x_7 < 10^{-2},$$
$$x_8 \in \{16, 32, 64, 128, 256\}, \text{ and}$$
$$10 \leq x_9 \leq 100.$$

These ranges cover common practice for CNN capacity, receptive-field control, regularization, and training dynamics in IDS settings. Because the search space mixes discrete and continuous variables and exhibits nonconvex, multimodal structure, we approach it with a population-based evolutionary optimizer.

C. Genetic Algorithm (GA) for hyperparameter optimization

We employ a Genetic Algorithm to solve the optimization, representing each candidate configuration as a chromosome that encodes the nine hyperparameters within the stated bounds. The process begins with a randomly initialized population of configurations. For each generation, every individual is instantiated as a 1D-CNN with the encoded hyperparameters, trained on the training split with early-stopping safeguards, and scored on a held-out validation set to produce its fitness $\phi(x)$. Parent selection favors higher-fitness individuals while preserving diversity; crossover then recombines parental genes to form offspring that inherit complementary architectural and training traits; mutation injects small random perturbations to selected genes to maintain exploration and reduce the risk of premature convergence. The next generation is formed by replacing the least fit individuals with offspring, optionally retaining a small elite set of top performers unchanged to stabilize progress. The algorithm terminates after a fixed number of generations or when the best fitness plateaus, returning the highest-fitness configuration for final training and test evaluation.

Algorithm 1: Optimizing 1D CNN hyperparameters for IDS using GA

Data: Population size $|P|$, Maximum number of generations G
Result: Best individual in P based on fitness ϕ
Initialize population P of size $|P|$ with random solutions
for $g = 1$ **to** G **do**
  for *each individual* x *in* P **do**
    Train 1D CNN model using hyperparameters x
    Evaluate fitness of x as $\phi(x)$ on the validation set
  end
  new_population ← empty set
  while *size of new_population* $< |P|$ **do**
    *parent*1, *parent*2 ← SELECTION(P)
    Crossover: $offspring1, offspring2$ ← CROSSOVER(*parent*1, *parent*2)
    Mutation: $offspring1$ ← MUTATE($offspring1$)
    $offspring2$ ← MUTATE($offspring2$)
    Add $offspring1$ and $offspring2$ to *new_population*
  end
  P ← *new_population*
  best_solution ← max(P, key=fitness)
  print "Generation:", g, "Best Solution:", *best_solution*
end
return best individual in P based on fitness ϕ
Function `Selection(P)`
Implement a selection strategy, e.g., roulette wheel or tournament
...
Function `Crossover(parent1, parent2)`
Implement a crossover strategy, e.g., single-point or uniform crossover
...
Function `Mutate(offspring)`
for *each hyperparameter* i *in* $offspring$ **do**
  With mutation probability:
    Randomly modify the value of i within its allowed range
end
return *mutated_offspring*

In practice, careful implementation details improve robustness and efficiency. Fitness is computed with a metric aligned to deployment goals (e.g., F1-score for imbalanced data), training uses consistent preprocessing and class balancing, and early stopping and capped epochs bound per-individual cost. Gene encodings ensure feasibility for discrete choices such as filter counts and kernel sizes, while continuous genes such as learning rate and dropout are clipped to their allowable intervals after crossover and mutation. This GA workflow operationalizes hyperparameter optimization for 1D-CNN-based IDS as a reproducible, end-to-end search over architecture and training settings, yielding

configurations that have been shown to substantially improve validation and test performance over non-optimized baselines on standard datasets.

D. Data Preprocessing

For this study, three widely used and diverse datasets have been selected to evaluate the performance of the proposed 1D-CNN-based network intrusion detection system (NIDS). These datasets have been chosen for their variety in terms of attack types, complexity, and relevance to the field of network security. Each dataset provides a unique challenge and contributes to a comprehensive understanding of how well the hyperparameter optimization strategy performs across different intrusion detection scenarios. The datasets used in this study include UNSW-NB15, CIC-IDS2017, and NSL-KDD.

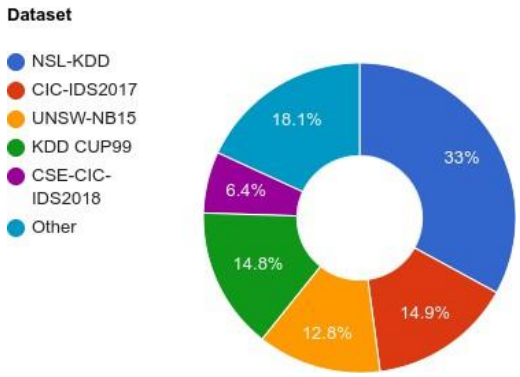


Figure 3. Percentage of datasets used in the CNN-IDS approaches

The **UNSW-NB15** dataset was developed by the University of New South Wales (UNSW) in collaboration with the Australian Centre for Cyber Security (ACCS) and aims to provide a more modern and diverse alternative to existing NIDS datasets. It contains a range of network traffic features and introduces a variety of attack categories, making it particularly suitable for testing advanced intrusion detection systems. The dataset includes attack types such as fuzzers, backdoors, denial-of-service (DoS), exploits, reconnaissance, shellcode, and worms. This broad selection of attack types offers a comprehensive test for the proposed 1D-CNN model, providing valuable insights into the model's ability to detect various contemporary attack patterns. All samples from the UNSW-NB15 dataset were utilized for the experiments in this study.

The **CIC-IDS2017** dataset, developed by the Canadian Institute for Cybersecurity (CIC), is one of the most comprehensive intrusion detection datasets available. It includes a wide range of attack scenarios, including brute force, Heartbleed, Botnet, DoS, DDoS, web attacks, and infiltration. This dataset provides labeled data for different types of network intrusions, which is extremely useful for training and testing machine learning-based models. The large and diverse set of features present in the CIC-IDS2017 dataset ensures that the model can be rigorously evaluated across a variety of intrusion types, making it an ideal choice for assessing the robustness of the hyperparameter optimization method. The entire dataset was employed in our experiments.

The **NSL-KDD** dataset is a refined version of the widely used KDD Cup 1999 dataset, which was designed to address several issues with the original dataset, such as redundant and duplicate records. NSL-KDD contains a mixture of normal traffic and various forms of intrusions, divided into four categories: DoS (Denial-of-Service), R2L (remote-to-local access), U2R (unauthorized access to local superuser privileges), and Probe (network surveillance and probing). This dataset includes both packet header features and content features derived from the payloads of network packets, offering a rich source of data for detecting various types of intrusions. Given its refined structure, NSL-KDD is particularly valuable for testing the generalization ability of the proposed 1D-CNN model in detecting a range of network threats.

The effectiveness of any machine learning model heavily depends on the quality of the data and the preprocessing steps applied. To ensure that the datasets are well-suited for training the proposed 1D-CNN, we performed extensive preprocessing on each of the datasets to clean and transform them into a format suitable for model training.

The first step in preprocessing involved checking for missing values or any null entries in the datasets. Missing or incomplete data can significantly skew model training, so any rows containing null values, infinite values, or NaNs were removed. This step ensures that the model is trained on valid, complete data.

We performed a correlation analysis to identify highly correlated features. Features with a correlation coefficient greater than 0.95 were excluded to reduce multicollinearity and ensure that each feature contributes unique information. This process is essential for improving model performance by eliminating redundancy.

Feature engineering is a key step to enhance the model's ability to make accurate predictions. In our study, we combined certain features to create new, more informative ones. For example, we combined the features "sbytes" and "dbytes" to generate a new feature, "network_bytes," which gives a more holistic view of the network traffic. Additionally, we excluded irrelevant features such as "srcip," "sport," "dstip," "dsport," and "attack_cat" as they did not provide value for the specific binary classification task.

To ensure that all features are on a similar scale, we applied **standardization** to the numerical features, transforming them to have a mean of 0 and a standard deviation of 1. This step prevents the model from being biased towards features with larger numerical values. For categorical variables like "proto," "service," and "state," we applied **one-hot encoding** to convert them into numerical values, which is necessary for machine learning algorithms to process categorical data effectively. This transformation resulted in a total of 197 features.

To rigorously evaluate the performance of our model, we split each dataset into three subsets: training, validation, and testing. The data was split in a 70:10:20 ratio, ensuring that the model is trained on a large portion of the data while being validated and tested on unseen examples. This strategy helps prevent overfitting and ensures that the model generalizes well to new data.

Each dataset required its own set of specific preprocessing steps. For the **CIC-IDS2017** dataset, we transformed the "Label" column to binary values, with "BENIGN" mapped to 0 and all other entries mapped to 1, indicating different types of intrusions. Irrelevant features like “Bwd PSH Flags” and “Fwd Avg Bytes/Bulk” were removed, and duplicate entries were excluded. We also applied feature scaling and split the dataset into training, validation, and testing subsets.

For the **NSL-KDD** dataset, we followed a similar approach, modifying the labels for binary classification and removing features like "num_outbound_cmds" that did not contribute valuable information. After one-hot encoding the categorical variables, we performed feature selection to keep only the most influential features. The dataset was then split and standardized as in the other datasets.

These preprocessing steps were designed to ensure that the data fed into the 1D-CNN model is clean, relevant, and standardized, thereby maximizing the model’s potential for learning and improving its detection performance. The next section outlines the experiments conducted with these preprocessed datasets to evaluate the efficacy of the hyperparameter optimization strategy.

5. EXPERIMENTS AND RESULTS

The proposed GA-optimized 1D-CNN-based intrusion-detection model was implemented in Python using the Keras library, built on top of TensorFlow. All experiments were conducted in this environment to ensure flexibility and reproducibility.

A. Evaluation Metrics

To assess the performance of the proposed model, we used four key evaluation metrics**: accuracy, precision, recall,** and **F1-score.** These metrics provide a balanced view of model performance, especially for binary classification problems such as intrusion detection.

1. **Precision (P)** measures how many of the predicted intrusions were actual attacks:

$$Precision = \frac{TP}{TP+FP}.$$

2. Recall (R) measures how many of the actual intrusions were correctly detected:

$$Recall = \frac{TP}{TP+FN}.$$

3. F1-score is the harmonic mean of precision and recall, providing a balanced assessment of both:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision+Recall}$$

4. Accuracy (A) measures the overall correctness of the model:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}.$$

These metrics together help quantify how well the model detects attacks while minimizing false alarms and missed detections.

B. Hyperparameter Optimization using Genetic Algorithm

In our experimental setup, the Genetic Algorithm (GA) was initialized with a population size of 20 and set to evolve over 10 generations. The mutation rate was fixed at 0.1, and the crossover rate at 0.5. The GA begins by generating an initial population of candidate hyperparameter sets for the 1D-CNN.

Each individual (i.e., a specific combination of hyperparameters) in the population was evaluated using a fitness function defined as the **validation accuracy** of the trained 1D-CNN. Based on their fitness scores, individuals with higher

performance were selected as parents for the next generation. The **crossover operation** produced offspring by mixing parent hyperparameters, while **mutation** introduced diversity by randomly altering certain hyperparameters within their valid ranges. The lowest-performing individuals were replaced by offspring, and this evolutionary process was repeated for 10 generations.

After the GA converged, the best individual—representing the optimal hyperparameter set—was used to train the final model. The optimized hyperparameters are summarized in **Table 1**.

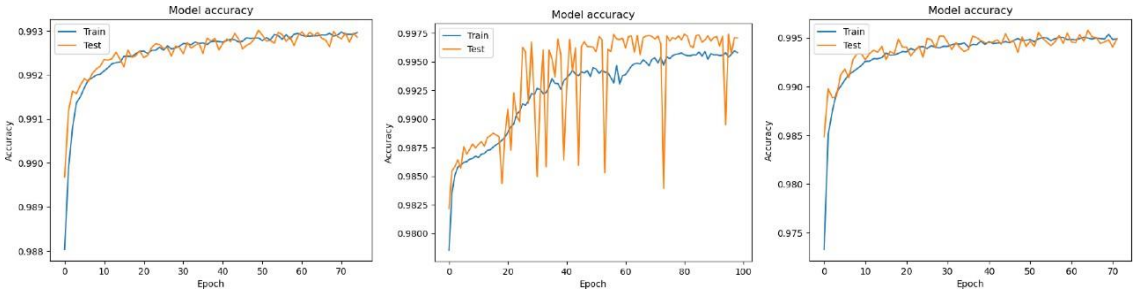
Table 1. GA-based hyperparameter optimization results

Hyperparameter	Range	UNSW-NB15	CIC-IDS2017	NSL-KDD
Number of filters	[16, 32, 64, 128, 256]	64, 128	64, 128	32, 64
Kernel size	[3, 5, 7, 9, 11]	5	9	9
Pooling size	(2, 6)	3	5	5
Number of dense layers	(1, 5)	2	2	1
Neurons in dense layers	[16, 32, 64, 128, 256]	128	256	128
Dropout rate	(0.1, 0.5)	0.277	0.114	0.253
Learning rate	$(1\times10^{-5}, 1\times10^{-2})$	0.0035	0.0012	0.0005
Batch size	[16, 32, 64, 128, 256]	256	64	32
Number of epochs	(10, 100)	75	99	72

After training and validation with these optimized hyperparameters, the final model was evaluated on the test sets of each dataset. The results are summarized in **Table 2**.

Table 2. Comparison of the effectiveness of GA on various datasets

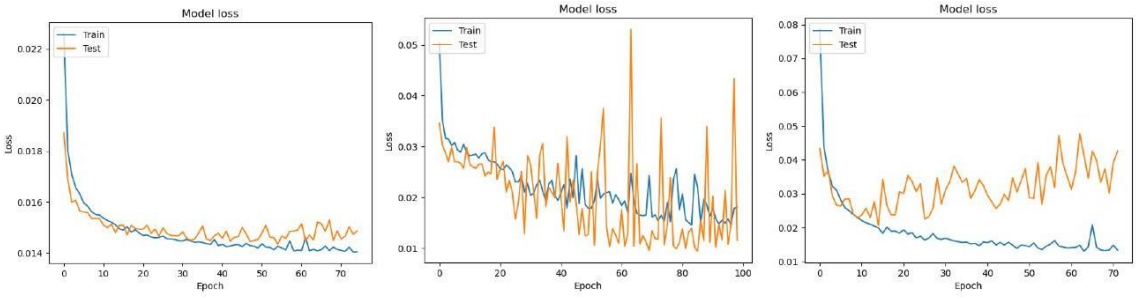
Dataset	Loss	Accuracy
UNSW-NB15	1.44	99.31%
CIC-IDS2017	1.15	99.71%
NSL-KDD	1.78	99.63%



(a) UNSW-NB15 (b) CIC-IDS2017 (c) NSL-KDD

Figure 4. Model accuracy when using GA

The accuracy and low loss values across all three datasets demonstrate the effectiveness of GA in optimizing hyperparameters for the 1D-CNN. Figures 4 and 5 illustrate the model's training and validation accuracy and loss trends, respectively, confirming that the GA-optimized configuration achieves fast convergence and stable learning performance.



(a) UNSW-NB15 (b) CIC-IDS2017 (c) NSL-KDD

Figure 5. Model loss when using GA

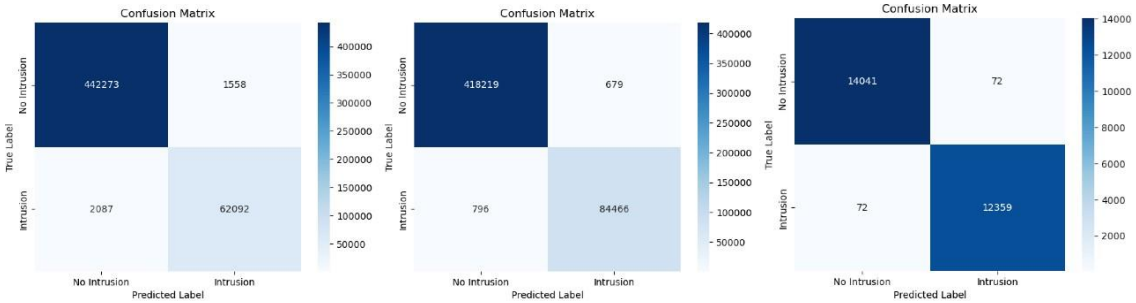
In summary, the GA-based optimization significantly improved the detection accuracy and reduced training loss across all datasets, validating the algorithm's robustness and its capability to efficiently explore the hyperparameter search space for complex intrusion detection tasks.

6. DISCUSSION

A. Analysis of the Results of the GA-Based Model

This section presents a detailed analysis of the results obtained using the Genetic Algorithm (GA) for hyperparameter optimization of the 1D-CNN-based intrusion-detection model. The primary goal of the GA was to identify the optimal set of hyperparameters that enhance the performance of the model across different benchmark datasets—UNSW-NB15, CIC-IDS2017, and NSL-KDD.

The GA-optimized model consistently achieved excellent results on all three datasets, with accuracies of 99.31 % on UNSW-NB15, 99.71 % on CIC-IDS2017, and 99.63 % on NSL-KDD. The corresponding low loss values (1.44, 1.15, and 1.78, respectively) indicate efficient convergence and stable learning. These findings clearly demonstrate the GA's capacity to fine-tune model hyperparameters effectively, thereby improving detection accuracy while maintaining low error rates.



(a) UNSW-NB15 (b) CIC-IDS2017 (c) NSL-KDD

Figure 6. Confusion matrix using GA

A deeper inspection of the confusion matrices (Figure 6) provides insights into the classification behavior of the GA-based model. On UNSW-NB15, the model accurately classifies 442,273 benign instances out of 444,831 and correctly identifies 62,092 intrusions, reflecting a high true-negative and true-positive rate. Similarly, on CIC-IDS2017, the model correctly recognizes 418,219 non-intrusive and 84,466 intrusive instances. On NSL-KDD, it identifies 12,359 intrusion cases accurately, confirming its robustness and adaptability across datasets.

Table 3. Comparative analysis of our models with the existing CNN-based models

Dataset	Model	Year	Accuracy	Precision	Recall	F1-Score
UNSW-NB15	CNN [22]	2018	94.9	-	-	-
	1D-CNN [23]	2019	91.2	87.53	96.17	91.59
	CNN-IDS [24]	2021	91	-	-	-
	BCNN [25]	2021	90.25	91	90	90.45
	GA-1D-CNN	**2023**	**99.31**	**99**	**98**	**98**
CIC-IDS2017	CNN-MCL [26]	2020	99.46	99.76	99.15	99.46
	CNN [27]	2022	99.41	-	-	-
	1D-CNN [28]	2022	98.68	99.2	98.94	98.96
	GA-1D-CNN	**2023**	**99.71**	**100**	**99**	**99**
NSL-KDD	CNN [29]	2018	80.13	-	-	-
	SMOTE-ENN [30]	2019	83.31	96.97	-	-
	CNN-1D [31]	2019	84.29	74.62	-	-
	Multi-CNN [32]	2019	86.95	89.56	87.25	88.41
	IBWNIDM [33]	2019	95.36	95.55	-	-
	Improved CNN [34]	2019	99.23	-	-	-
	IDS-CNN [35]	2020	97.7	-	-	-
	DMCNN [36]	2020	94.65	96.66	-	-
	CNN(AVG) [37]	2020	88.82	-	-	90.67
	AS-CNN [38]	2020	84.08	80	-	-
	BCNN-DFS [25]	2021	90.14	90	90	90
	CNN [39]	2021	99	98	97	97
	CNN-B [27]	2022	84.82	85.74	87.96	-
	GA-1D-CNN	**2023**	**99.63**	**99**	**99**	**99**

Overall, the GA-optimized 1D-CNN demonstrates remarkable sensitivity and specificity, achieving a delicate balance between detecting malicious activities and minimizing false alarms. The evolutionary search mechanism of GA—combining selection, crossover, and mutation—proves highly effective in exploring complex, nonlinear hyperparameter

spaces and avoiding local minima. This ensures consistent discovery of near-optimal solutions for diverse intrusion-detection tasks.

B. Comparison with Existing Methods

To assess the contribution of the proposed GA-optimized model, we compare its performance with previously reported CNN-based intrusion-detection systems. The results (Table 3) clearly show that the GA-1D-CNN surpasses conventional CNN architectures that rely on manually or heuristically selected hyperparameters.

Existing studies often overlook the critical influence of hyperparameter tuning on model performance. By integrating a systematic evolutionary approach, our GA-based method overcomes this limitation and consistently achieves higher precision, recall, and F1-scores across all datasets. The fine-tuning process allows the CNN to automatically adapt its structure and learning dynamics to the intrinsic properties of each dataset, leading to enhanced generalization and detection accuracy.

Table 4. Comparison of our models with hybrid CNN and ML IDS

Dataset	Model	Year	Accuracy	Precision	Recall	F1-Score
UNSW-NB15	SGM-CNN [40]	2020	98.82	99.74	-	95.53
	OCNN-HMLSTM [6]	2021	96.33	100	95.87	98.13
	CNN + LSTM [41]	2022	87.6	85.5	90.6	88
	ODODL-IDS [10]	2022	92.87	97.33	77.53	72.53
	CNN + LSTM [42]	2023	93.21	-	-	-
	CNN-LSTM [43]	2023	96.99	95.45	-	-
	OHDNN + ECRF [44]	2023	98.3	97.5	96.7	97.1
	GA-1D-CNN	**2023**	**99.31**	**99**	**98**	**98**
CIC-IDS2017	SDCNN [45]	2021	99.35	-	-	-
	Tree-CNN [46]	2021	98	-	-	98
	ODODL-IDS [10]	2022	97.62	97.26	97.25	99
	CNN 1D + BLSTM [47]	2023	98	86	84	81
	GAN-CNN-BiLSTM [48]	2023	96.32	96.55	95.38	96.04
	GA-1D-CNN	**2023**	**99.71**	**100**	**99**	**99**
NSL-KDD	DCNN [4]	2018	85.22	97	-	-
	IFS-CNN-BG [49]	2020	98.24	95.44	-	-
	PSO-CCNN [50]	2021	98.71	-	-	-
	OCNN-HMLSTM [6]	2021	90.67	86.71	95.19	91.46
	CNN + LSTM [41]	2022	95.2	99.5	90.8	94.9
	ODODL-IDS [10]	2022	89.09	95.38	99.65	78.44
	CNN-LSTM [43]	2023	97.23	96.45	-	-
	OHDNN [44]	2023	97.17	97.32	97.02	95.92
	TL-CNN-IDS [11]	2023	99.53	97.63	96.77	97.13
	LSTM-CNN [5]	2023	97.8	93.71	96.19	95.46
	GA-1D-CNN	**2023**	**99.63**	**99**	**99**	**99**

Compared with hybrid CNN-ML intrusion-detection frameworks (Table 4), the GA-optimized model maintains competitive or superior results while preserving architectural simplicity. Rather than combining multiple models, the proposed approach achieves excellence through intelligent hyperparameter selection, highlighting the practical power of evolutionary optimization for IDS design.

C. Strengths and Limitations

The GA-based 1D-CNN approach offers several significant strengths:

First, it demonstrates superior accuracy and reliability across multiple benchmark datasets, confirming its robustness and adaptability to varying network conditions and attack patterns.

Second, it achieves high precision and recall, ensuring effective intrusion detection with minimal false positives and negatives—an essential characteristic for operational IDS deployment.

Third, the systematic hyperparameter optimization using GA allows the model to discover the most efficient architecture and learning configuration without manual intervention. This reduces human bias and enhances reproducibility.

Despite these advantages, certain limitations exist. The GA introduces computational overhead due to its iterative nature and the need to train multiple models during optimization. Although this cost is offset by the performance gains, practical implementations may require parallelization or cloud-based computing resources. Additionally, as with most machine-learning systems, performance can be dataset-dependent, meaning that optimal hyperparameters for one network environment may not directly transfer to another. Lastly, class imbalance in IDS datasets remains a persistent challenge. While the GA-optimized model performs well under such conditions, integrating techniques like data resampling or cost-sensitive learning could further strengthen detection of minority attack classes.

7. CONCLUSION

This study presented a Genetic Algorithm-based framework for hyperparameter optimization in one-dimensional convolutional neural-network models for network intrusion detection. By systematically exploring the hyperparameter search space through evolutionary principles of selection, crossover, and mutation, the GA efficiently identified near-optimal configurations that significantly enhanced model performance across multiple benchmark datasets.

The proposed GA-optimized 1D-CNN achieved outstanding accuracy, precision, recall, and F1-scores on the UNSW-NB15, CIC-IDS2017, and NSL-KDD datasets, outperforming conventional CNN-based intrusion-detection models that rely on manually tuned parameters. These results underscore the critical role of automated hyperparameter optimization in improving the reliability and robustness of deep-learning-based intrusion-detection systems.

In addition to its high detection capability, the GA-optimized model demonstrated strong generalization and adaptability across diverse network environments and attack categories. This highlights its practical applicability in real-world cybersecurity contexts, where attack patterns and traffic distributions constantly evolve.

Despite its effectiveness, the GA-based optimization approach introduces additional computational overhead due to iterative model training. Future work may address this limitation by incorporating parallel GA implementations or hybrid optimization strategies that combine GA with other search techniques to accelerate convergence. Furthermore, extending the model to multiclass classification and online learning scenarios could enhance its operational value in dynamic network settings.

In summary, the research confirms that Genetic Algorithm-driven hyperparameter optimization offers a powerful and flexible solution for designing next-generation, high-performance intrusion-detection systems. By uniting the adaptability of evolutionary computation with the learning capacity of deep neural networks, this approach contributes a significant advancement toward more secure, intelligent, and resilient network-defense mechanisms.

REFERENCES

[1] Kilichev, D.; Kim, W. Hyperparameter Optimization for 1D-CNN-Based Network Intrusion Detection Using GA and PSO. *Mathematics* **2023**, *11*, 3724. https://doi.org/10.3390/math11173724

[2] Kilichev, D.; Turimov, D.; Kim, W. Next–Generation Intrusion Detection for IoT EVCS: Integrating CNN, LSTM, and GRU Models. *Mathematics* **2024**, *12*, 571. https://doi.org/10.3390/math12040571

[3] Makhmudov, F.; Kilichev, D.; Giyosov, U.; Akhmedov, F. Online Machine Learning for Intrusion Detection in Electric Vehicle Charging Systems. *Mathematics* **2025**, *13*, 712. https://doi.org/10.3390/math13050712

[4] A. Abdusalomov, D. Kilichev, R. Nasimov, I. Rakhmatullayev and Y. I. Cho, "Optimizing Smart Home Intrusion Detection with Harmony-Enhanced Extra Trees," in IEEE Access, doi: 10.1109/ACCESS.2024.3422999

[5] Selvarajan, P.; Salman, R.; Ahamed, S.; Jayasuriya, P. Networks Intrusion Detection Using Optimized Hybrid Network. In Proceedings of the 2023 International Conference on Smart Computing and Application (ICSCA), Hail, Saudi Arabia, 5–6 February 2023; pp. 1–6.

[6] Kanna, R.P.; Santhi, P. Unified Deep Learning approach for Efficient Intrusion Detection System using Integrated Spatial–Temporal Features. Knowl.-Based Syst. 2021, 226, 107132.

[7] Zhao, X.; Su, H.; Sun, Z. An Intrusion Detection System Based on Genetic Algorithm for Software-Defined Networks. Mathematics 2022, 10, 3941.

[8] Yang, L.; Shami, A. A Transfer Learning and Optimized CNN Based Intrusion Detection System for Internet of Vehicles. In Proceedings of the ICC 2022—IEEE International Conference on Communications, Foshan, China, 11–13 August 2022; pp. 2774–2779.

[9] Chen, Y.; Lin, Q.; Wei, W.; Ji, J.; Wong, K.C.; Coello, C.A. Intrusion detection using multi-objective evolutionary convolutional neural network for Internet of Things in Fog computing. Knowl.-Based Syst. 2022, 244, 108505.

[10] Ragab, M.; Sabir, F. Outlier detection with optimal hybrid deep learning enabled intrusion detection system for ubiquitous and smart environment. Sustain. Energy Technol. Assess. 2022, 52, 102311.

[11] Yan, F.; Zhang, G.; Zhang, D.; Sun, X.; Hou, B.; Yu, N. TL-CNN-IDS: Transfer learning-based intrusion detection system using convolutional neural network. J. Supercomput. 2023, 242
[12] Okey, O.D.; Melgarejo, D.C.; Saadi, M.; Rosa, R.L.; Kleinschmidt, J.H.; Rodríguez, D.Z. Transfer Learning Approach to IDS on Cloud IoT Devices Using Optimized CNN. IEEE Access 2023, 11, 1023–1038.
[13] El-Ghamry, A.; Darwish, A.; Hassanien, A.E. An optimized CNN-based intrusion detection system for reducing risks in smart farming. Internet Things 2023, 22, 100709.
[14] Rosay, A.; Riou, K.; Carlier, F.; Leroux, P. Multi-layer perceptron for network intrusion detection: From a study on two recent data sets to deployment on automotive processor. Ann. Telecommun. 2022, 77, 371–394.
[15] Obeidat, A.; Yaqbeh, R. Smart Approach for Botnet Detection Based on Network Traffic Analysis. J. Electr. Comput. Eng. 2022, 2022, 3073932.
[16] Zhang, X.; Zou, D.; Shen, X. A Novel Simple Particle Swarm Optimization Algorithm for Global Optimization. Mathematics 2018, 6, 287.
[17] Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; pp. 1–6.
[18] Sharafaldin, I.; Habibi Lashkari, A.; Ghorbani, A.A. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy—ICISSP, Funchal, Portugal, 22–24 January 2018; pp. 108–116.
[19] Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
[20] Gautam, S.; Henry, A.; Zuhair, M.; Rashid, M.; Javed, A.R.; Maddikunta, P.K.R. A Composite Approach of Intrusion Detection Systems: Hybrid RNN and Correlation-Based Feature Optimization. Electronics 2022, 11, 3529.
[21] Preuveneers, D.; Rimmer, V.; Tsingenopoulos, I.; Spooren, J.; Joosen, W.; Ilie-Zudor, E. Chained Anomaly Detection Models for Federated Learning: An Intrusion Detection Case Study. Appl. Sci. 2018, 8, 2663.
[22] Potluri, S.; Ahmed, S.; Diedrich, C. Convolutional Neural Networks for Multi-class Intrusion Detection System. In Proceedings of the Mining Intelligence and Knowledge Exploration, Cluj-Napoca, Romania, 20–22 December 2018; Groza, A., Prasath, R., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 225–238.
[23] Azizjon, M.; Jumabek, A.; Kim, W. 1D CNN based network intrusion detection with normalization on imbalanced data. In Proceedings of the 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Fukuoka, Japan, 19–21 February 2020; pp. 218–224.
[24] Gamal, M.; Abbas, H.M.; Moustafa, N.; Sitnikova, E.; Sadek, R.A. Few-Shot Learning for Discovering Anomalous Behaviors in Edge Networks. Comput. Mater. Contin. 2021, 69, 1823–1837.
[25] Al-Turaiki, I.; Altwaijry, N. A Convolutional Neural Network for Improved Anomaly-Based Network Intrusion Detection. Big Data 2021, 9, 233–252.
[26] Mohammadpour, L.; Ling, T.C.; Liew, C.S.; Aryanfar, A. A mean convolutional layer for intrusion detection system. Secur. Commun. Netw. 2020, 2020, 8891185.
[27] Aldarwbi, M.Y.; Lashkari, A.H.; Ghorbani, A.A. The sound of intrusion: A novel network intrusion detection system. Comput. Electr. Eng. 2022, 104, 108455.
[28] Qazi, E.U.H.; Almorjan, A.; Zia, T. A One-Dimensional Convolutional Neural Network (1D-CNN) Based Deep Learning System for Network Intrusion Detection. Appl. Sci. 2022, 12, 7986.
[29] Ding, Y.; Zhai, Y. Intrusion detection system for NSL-KDD dataset using convolutional neural networks. In Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence, Shenzhen, China, 8–10 December 2018; pp. 81–85.
[30] Zhang, X.; Ran, J.; Mi, J. An Intrusion Detection System Based on Convolutional Neural Network for Imbalanced Network Traffic. In Proceedings of the 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 19–20 October 2019; pp. 456–460.
[31] Verma, A.K.; Kaushik, P.; Shrivastava, G. A Network Intrusion Detection Approach Using Variant of Convolution Neural Network. In Proceedings of the 2019 International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 17–19 July 2019; pp. 409–416.
[32] Li, Y.; Xu, Y.; Liu, Z.; Hou, H.; Zheng, Y.; Xin, Y.; Zhao, Y.; Cui, L. Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. Measurement 2020, 154, 107450.
[33] Yang, H.; Wang, F. Wireless Network Intrusion Detection Based on Improved Convolutional Neural Network. IEEE Access 2019, 7, 64366–64374.
[34] Khan, R.U.; Zhang, X.; Alazab, M.; Kumar, R. An Improved Convolutional Neural Network Model for Intrusion Detection in Networks. In Proceedings of the 2019 Cybersecurity and Cyberforensics Conference (CCC), Melbourne, Australia, 8–9 May 2019; pp. 74–77.
[35] Wang, H.; Cao, Z.; Hong, B. A network intrusion detection system based on convolutional neural network. J. Intell. Fuzzy Syst. 2020, 38, 7623–7637.

[36] Wang, X.; Yin, S.; Li, H.; Wang, J.; Teng, L. A network intrusion detection method based on deep multi-scale convolutional neural network. Int. J. Wirel. Inf. Netw. 2020, 27, 503–517.
[37] Jo, W.; Kim, S.; Lee, C.; Shon, T. Packet Preprocessing in CNN-Based Network Intrusion Detection System. Electronics 2020, 9, 1151.
[38] Hu, Z.; Wang, L.; Qi, L.; Li, Y.; Yang, W. A Novel Wireless Network Intrusion Detection Method Based on Adaptive Synthetic Sampling and an Improved Convolutional Neural Network. IEEE Access 2020, 8, 195741–195751.
[39] Akhtar, M.S.; Feng, T. Deep learning-based framework for the detection of cyberattack using feature engineering. Secur. Commun. Netw. 2021, 2021, 6129210.
[40] Zhang, H.; Huang, L.; Wu, C.Q.; Li, Z. An effective convolutional neural network based on SMOTE and Gaussian mixture model for intrusion detection in imbalanced dataset. Comput. Netw. 2020, 177, 107315.
[41] Meliboev, A.; Alikhanov, J.; Kim, W. Performance Evaluation of Deep Learning Based Network Intrusion Detection System across Multiple Balanced and Imbalanced Datasets. Electronics 2022, 11, 515.
[42] Altunay, H.C.; Albayrak, Z. A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks. Eng. Sci. Technol. Int. J. 2023, 38, 101322.
[43] Thilagam, T.; Aruna, R. LM-GA: A Novel IDS with AES and Machine Learning Architecture for Enhanced Cloud Storage Security. J. Mach. Comput. 2023, 3, 69–79.
[44] Karthic, S.; Kumar, S.M. Hybrid Optimized Deep Neural Network with Enhanced Conditional Random Field Based Intrusion Detection on Wireless Sensor Network. Neural Process. Lett. 2023, 55, 459–479.
[45] Khan, A.S.; Ahmad, Z.; Abdullah, J.; Ahmad, F. A Spectrogram Image-Based Network Anomaly Detection System Using Deep Convolutional Neural Network. IEEE Access 2021, 9, 87079–87093.
[46] Mendonça, R.V.; Teodoro, A.A.M.; Rosa, R.L.; Saadi, M.; Melgarejo, D.C.; Nardelli, P.H.J.; Rodríguez, D.Z. Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network. IEEE Access 2021, 9, 61024–61034.
[47] Bowen, B.; Chennamaneni, A.; Goulart, A.; Lin, D. BLoCNet: A hybrid, dataset-independent intrusion detection system using deep learning. Int. J. Inf. Secur. 2023, 22, 893–917.
[48] Li, S.; Li, Q.; Li, M. A Method for Network Intrusion Detection Based on GAN-CNN-BiLSTM. Int. J. Adv. Comput. Sci. Appl. 2023, 14, 507–515.
[49] Nguyen, M.T.; Kim, K. Genetic convolutional neural network for intrusion detection systems. Future Gener. Comput. Syst. 2020, 113, 418–427.
[50] Bhuvaneshwari, K.S.; Venkatachalam, K.; Hubálovský, S.; Trojovský, P.; Prabu, P. Improved Dragonfly Optimizer for Intrusion Detection Using Deep Clustering CNN-PSO Classifier. *Comput. Mater. Contin.* **2022**, *70*, 5949–5965.