

Large-Scale Binary Network Intrusion Classification Using a 1D Convolutional Neural Network on CSE-CIC-IDS2018: Performance Evaluation and Methodological Audit

Sarah Yasmin Yuszairulakhbar¹, Ahmad Kamalrulzaman Othman^{2*}, Yusnita Sokman²

¹Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Shah Alam, Bangunan Al-Khawarizmi, 40450 Shah Alam, Selangor Darul Ehsan, Malaysia

²Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA Cawangan Johor, Kampus Segamat, 85000 Segamat, Johor, Malaysia

*Corresponding author

DOI: <https://doi.org/10.51584/IJRIAS.2026.11080099>

Received: 27 August 2026; Accepted: 01 September 2026; Published: 11 September 2026

ABSTRACT

Deep learning is increasingly applied to network intrusion detection because it can learn discriminative representations from high-dimensional flow records. This study evaluates a one-dimensional convolutional neural network (1D-CNN) for binary classification of benign and malicious traffic using CSE-CIC-IDS2018 and simultaneously audits the experimental pipeline for threats to independent generalization. The evaluated workflow processed the dataset in chunks, selected traffic features using mutual information, engineered packet-rate and flag-ratio variables, reduced the representation to 20 predictors, balanced the training subset using SMOTE, and trained a regularized Conv1D model with batch normalization, max pooling, dropout, AdamW optimization, and early stopping. The classifier used three convolutional stages with 32, 64, and 128 filters, followed by a regularized 128-unit dense layer and sigmoid output. This architecture was retained consistently throughout the reported benchmark experiment. For the selected 80:20 experiment, the reported confusion matrix contains 1,223,435 true negatives, 45,415 false positives, 29,948 false negatives, and 273,224 true positives. Recalculation gives 95.206% accuracy, 96.421% specificity, 85.747% precision, 90.122% recall, and an F1-score of 87.880%, while the reported ROC-AUC is 97.47%. The methodological audit identifies feature selection and scaling before data partitioning, use of the nominal test partition for validation-based model selection, chunk-wise categorical label coding, and inconsistent post-processing sample counts as important threats to confirmatory interpretation. The findings provide useful benchmark evidence but do not establish leakage-controlled generalization. A stronger replication should split observations before learned preprocessing, map labels deterministically, isolate validation from a locked test set, repeat training across multiple random seeds, and add chronological or cross-dataset evaluation when deployment-level claims are intended.

Keywords: network intrusion detection, 1D-CNN, CSE-CIC-IDS2018, data leakage, binary classification

INTRODUCTION

Network intrusion detection remains difficult because contemporary networks combine high traffic volume, heterogeneous protocols, rapidly changing services, and attack behaviour that does not remain fixed. Signature-oriented intrusion detection can identify known patterns efficiently, but it is less suitable when traffic departs from predefined rules or when analysts face a large volume of alerts. Data-driven network intrusion detection has consequently moved toward machine learning and deep learning models that infer statistical relationships from labelled traffic flows [1,2]. A trained classifier can screen large collections of flow records automatically, but the credibility of the result depends as much on evaluation design as on model architecture.

Convolutional neural networks are increasingly used in this setting. Although convolution was developed for structured local patterns, one-dimensional convolution can be applied to ordered vectors of network-flow attributes. Filters then learn local combinations of traffic characteristics, while pooling and dense layers convert these learned representations into class decisions. Published CNN-based IDS studies report strong predictive performance, but comparison across papers remains difficult because studies use different datasets, class definitions, preprocessing pipelines, train-test protocols, and evaluation metrics [4]. The same architecture can therefore appear stronger or weaker depending on the experimental design around it.

Machine learning and deep learning for intrusion detection

Machine learning-based NIDS research spans linear classifiers, tree ensembles, recurrent networks, convolutional networks, and hybrid architectures. Surveys repeatedly identify class imbalance, high dimensionality, dataset realism, and poor cross-environment generalization as recurring problems [1,2]. These issues are connected. A classifier trained on a benchmark with dominant benign traffic can achieve high accuracy while missing minority attacks, and a model that exploits benchmark-specific artefacts can perform strongly on a random holdout but deteriorate when the operational distribution changes.

CNN-based IDS research attempts to reduce reliance on manually specified decision rules by learning intermediate representations. Existing work has used CNNs as end-to-end classifiers, feature extractors, or components of hybrid models. Qazi et al. reported strong multiclass performance with a 1D-CNN on CICIDS2017 [5]. Wang et al. compared several deep architectures on CSE-CIC-IDS2018 and also highlighted the computational cost of more complex hybrid models [6]. Wang and Ghaleb introduced attention into a CNN for the same benchmark [7], while more recent work has continued to investigate lightweight 1D-CNN designs and the balance between discrimination and computational efficiency [8]. These studies establish 1D convolution as a viable direction, but they do not remove the need for leakage-controlled evaluation.

The wider literature represented in the source manuscript also includes real-time deep-learning intrusion detection, CNN-based IDS designs, experimental machine-learning comparisons, adaptive synthetic sampling with improved convolutional models, broad AI/ML NIDS studies, feature-optimization frameworks, and adaptive ensemble approaches [12-18]. This breadth shows that model families and preprocessing strategies vary substantially across published work. For that reason, a defensible comparison should prioritize models trained on the same data partition and preprocessing protocol rather than infer superiority from cross-paper accuracy values obtained under incompatible experimental conditions.

Class imbalance, feature selection, and evaluation credibility

Network traffic datasets are rarely balanced. Benign observations may dominate some capture periods, while particular attacks dominate others. Accuracy alone can therefore obscure poor minority-class behaviour. Precision, recall, F1-score, specificity, false-positive rate, and ROC-AUC provide a more informative view of binary intrusion performance. Operationally, false positives consume analyst attention and false negatives represent malicious flows that remain undetected. The preferred trade-off is therefore a deployment decision rather than a purely mathematical one.

Feature selection can reduce dimensionality and computation, but supervised selection becomes a leakage channel if the selection criterion sees the full dataset before the test set is isolated. The evaluated implementation uses mutual information through SelectKBest, which depends on the target labels. Performing this step before partitioning allows information from the future test labels to influence which predictors reach the model. Recent work has explicitly examined the inflation that can result from feature-selection leakage [9]. Leakage-safe protocols instead fit selectors and scalars only on training data and then apply the learned transformations unchanged to validation and test partitions [11].

Evaluation integrity is particularly important for CSE-CIC-IDS2018 because its scale can create an impression of realism even when the split remains internal to the same capture process. Random row-level splitting assumes sufficient independence between training and test observations. In network data, that assumption can fail when flows from the same attack campaign, host, time period, or traffic generator appear in both partitions.

Reliable NIDS evaluation therefore requires careful control of data preparation and metric interpretation [3]. Cross-evaluation provides a stronger test because it exposes the model to a distribution not used during fitting and can reveal weaknesses hidden by a single-dataset split [10].

Study objective and contribution

This paper has two objectives. First, it consolidates the source experiment into a reproducible empirical account of a large-scale 1D-CNN binary intrusion classifier. Second, it audits the experimental pipeline against current evaluation standards and separates observed benchmark performance from claims that require leakage-controlled replication. The contribution is therefore methodological as well as predictive. The paper recomputes the principal confusion-matrix metrics, identifies publication-relevant weaknesses in the implementation order, and specifies a corrected protocol that preserves the original scientific question while strengthening the independence of the final estimate.

MATERIALS AND METHODS

Dataset and binary target construction

The study used the publicly available CSE-CIC-IDS2018 network intrusion dataset. The dataset representation includes ten CSV files collected on different dates, with benign traffic and attack labels including FTP brute force, DoS GoldenEye, DoS Hulk, DDoS LOIC-HTTP, DDoS HOIC, web brute force, infiltration, and bot traffic. Approximately 80 columns represent flow duration, packet counts, byte counts, inter-arrival statistics, TCP flag counts, active and idle periods, protocol information, timestamps, and the class label.

Table 1. Flow-feature groups represented in the dataset and modelling pipeline.

Feature group	Examples	Role in intrusion classification
Flow identity and protocol	Destination port, protocol	Represent service and protocol context
Traffic volume	Forward/backward packets, total bytes	Represent flow magnitude and directionality
Packet-length statistics	Mean, minimum, maximum, variance	Capture packet-size behaviour
Inter-arrival timing	Flow IAT, forward IAT, backward IAT	Represent temporal behaviour within flows
TCP flags	SYN, ACK, FIN, RST counts	Capture connection-state patterns
Activity statistics	Active mean, idle mean	Represent burst and idle structure
Engineered rates	Forward rate, backward rate, flag ratio	Summarize normalized flow intensity

The modelling task was simplified from multiclass attack recognition to binary classification. In the source implementation, text labels were first transformed to categorical integer codes separately within each input chunk, after which code 0 was interpreted as benign and any nonzero code as malicious. This procedure is unsafe because categorical coding can differ across chunks. A chunk containing only attack records may assign code 0 to an attack category. A confirmatory replication should map the original text label explicitly, for example by assigning benign only when the original label equals the known benign string, before any chunk-specific categorical encoding occurs.

The source experiment reports approximately 16 million total records and an 80:20 split of approximately 12.8 million training and 3.2 million testing observations. The selected confusion matrix, however, contains 1,572,022 evaluated test observations. Because the confusion-matrix entries provide exact class counts, this paper uses 1,572,022 as the effective test-set size for metric recomputation and treats the larger approximate test count as an unresolved reporting discrepancy. A reproducible rerun should log the row count and class distribution after every cleaning and transformation stage.

Preprocessing and feature engineering

The source implementation processed CSV files in chunks of 10,000 rows to limit memory pressure. Completely empty columns were removed, duplicate rows were dropped within each chunk, and categorical missing values were imputed with the most frequent value. Protocol and label variables were encoded numerically. Timestamp information was converted into date, hour, minute, and second fields. Numeric variables were then transformed with MinMaxScaler using a partial-fit procedure across the complete cleaned dataset.

A second modelling script selected a smaller set of base and additional features, then derived forward packet rate, backward packet rate, and a SYN flag ratio. The final preselection input contained 23 variables. Mutual-information SelectKBest reduced these to 20 predictors. StandardScaler then standardized the selected matrix. Only after these learned transformations were fitted did the implementation create a stratified 80:20 train-test split with random state 42. SMOTE was applied only to the training subset, which is the correct location for resampling. The balanced training data and untouched test observations were reshaped to samples by features by one channel for Conv1D input.

Two details require correction in a confirmatory rerun. First, SelectKBest and StandardScaler were fitted before train-test partitioning, which allows test information to influence selected predictors and scaling parameters. Second, MinMaxScaler had already been fitted to the complete dataset in the earlier preprocessing stage, creating an additional path for the future test distribution to affect training representations. The corrected procedure must split cleaned observations first and fit every learned transformation only on the training portion.

The engineered packet-rate variables also require careful interpretation. In the source workflow they are computed after the initial global scaling stage, so the ratio no longer corresponds directly to a packet-per-time quantity in the original units. A reproducible pipeline should calculate row-wise rate features from raw or unit-preserved variables before statistical scaling. It should also guard against zero or near-zero flow durations and record how infinities, missing values, and extreme rates are handled.

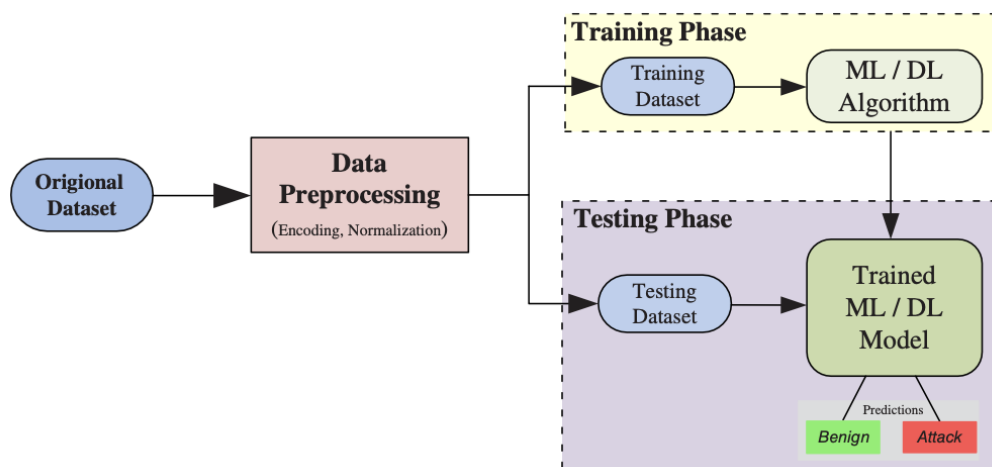


Figure 1. Generic training and testing organization for machine-learning intrusion detection, adapted from the source study framework.

CNN architecture and optimization

The classifier was implemented as a one-dimensional CNN. The input first passed through Gaussian noise with standard deviation 0.1 as a regularization mechanism. Three Conv1D layers used 32, 64, and 128 filters, each with kernel size 3 and ReLU activation. Batch normalization followed each convolutional stage. Max pooling reduced the feature dimension after the first and third convolutional blocks. The resulting representation was flattened and passed to a 128-unit dense layer with ReLU activation and L2 regularization of 0.01. Dropout with probability 0.5 preceded the final sigmoid output for binary prediction.

Table 2. CNN architecture reconstructed from the evaluated implementation.

Stage	Configuration	Purpose
Input	20 selected features x 1 channel	Represent tabular flow vector for Conv1D
Regularization	GaussianNoise(0.1)	Reduce sensitivity to small perturbations
Conv block 1	32 filters, kernel 3, ReLU + BatchNorm + MaxPool	Learn low-level feature combinations
Conv block 2	64 filters, kernel 3, ReLU + BatchNorm	Increase representation capacity
Conv block 3	128 filters, kernel 3, ReLU + BatchNorm + MaxPool	Learn higher-level combinations
Dense	128 ReLU, L2=0.01, Dropout=0.5	Regularized nonlinear decision layer
Output	1 sigmoid unit	Binary benign/malicious probability
Optimizer	AdamW, learning rate 0.001	Gradient-based parameter update

AdamW with an initial learning rate of 0.001 optimized binary cross-entropy. The model tracked accuracy, AUC, precision, and recall. Training allowed up to 50 epochs with batch size 128. Early stopping monitored validation recall with patience 7, ReduceLROnPlateau monitored validation loss, and ModelCheckpoint saved the model with the best validation recall. These mechanisms are reasonable for controlling training duration, but the implementation supplied the nominal test set as validation data. Model selection was therefore informed by test labels, which removes the independence required for a final holdout evaluation.

A stronger design should divide the training portion again into development and validation subsets or use stratified cross-validation inside the training set. Early stopping, learning-rate adjustment, threshold selection, and checkpoint choice should operate only on internal validation data. The final test set should remain untouched until the preprocessing configuration, architecture, stopping rule, and decision threshold are fixed.

Evaluation metrics and metric audit

Malicious traffic was treated as the positive class. Accuracy was calculated as $(TP+TN)/N$, specificity as $TN/(TN+FP)$, precision as $TP/(TP+FP)$, recall as $TP/(TP+FN)$, and F1 as the harmonic mean of precision and recall. The selected 80:20 table reports $TN=1,223,435$, $FP=45,415$, $FN=29,948$, and $TP=273,224$. These values sum to 1,572,022 and permit independent metric recomputation. Recalculation produces 95.206% accuracy, 96.421% specificity, 85.747% precision, 90.122% recall, and 87.880% F1. The source report gives the same accuracy, precision, recall, and F1 after rounding but reports specificity as 97.61%. The confusion-matrix counts support 96.421%, so the recomputed value is used here.

RESULTS

Selected 80:20 configuration

Table 3. Selected 80:20 confusion matrix and recomputed binary-classification metrics.

Measure	Value
True negatives (TN)	1,223,435
False positives (FP)	45,415
False negatives (FN)	29,948
True positives (TP)	273,224
Accuracy	95.206%

Measure	Value
Specificity	96.421%
Precision	85.747%
Recall	90.122%
F1-score	87.880%
Reported ROC-AUC	97.47%

The selected configuration correctly classified 1,496,659 of the 1,572,022 observations represented by the confusion matrix. The error structure is asymmetric. False positives number 45,415, while false negatives number 29,948. Because the benign class is much larger than the malicious class, the false-positive count remains substantial even though specificity exceeds 96%. Operationally, every false positive can become an unnecessary alert, while every false negative represents a malicious flow that is not detected.

Precision is lower than recall. The classifier identifies about 90.1% of malicious observations, but only about 85.7% of observations predicted as malicious are truly malicious. This pattern indicates a model that favours detection sensitivity over alert purity at the default threshold. Whether that trade-off is appropriate depends on the relative cost of missed attacks and analyst workload. The reported ROC-AUC of 97.47% suggests strong ranking discrimination across thresholds, but the source material does not provide the underlying ROC points or a confidence interval, so further threshold-specific claims are not supported.

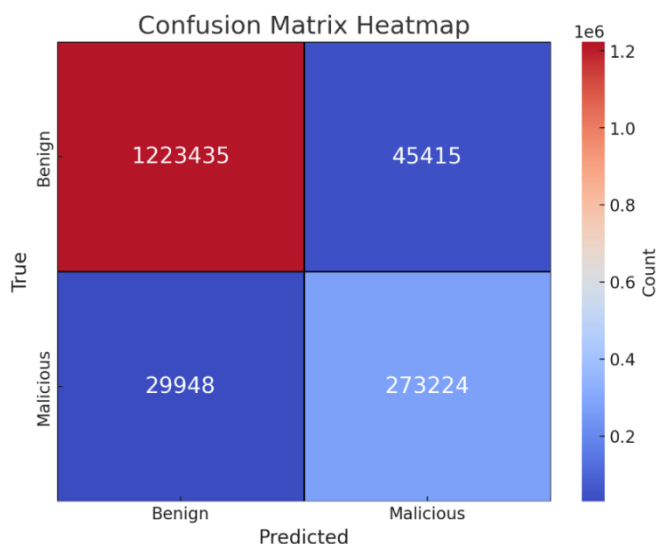


Figure 2. Confusion-matrix heatmap for the selected 80:20 experiment.

Training behaviour

The source experiment includes training and validation curves from a run that stopped after 14 epochs under an early-stopping rule. Training accuracy rises gradually, while validation accuracy fluctuates more strongly during early epochs before stabilizing. Training loss declines over time, whereas validation loss temporarily increases before settling in later epochs. The qualitative pattern is compatible with convergence without obvious runaway training loss, but interpretation is limited because the nominal test partition served as validation data. The stopping behaviour also shows why a maximum epoch setting is not equivalent to actual training duration. A confirmatory replication should log the stopping epoch, best checkpoint epoch, wall-clock training time, and validation criterion for every run.

Internal consistency audit

The internal audit identifies several issues that are publication-relevant because they affect either the independence of the evaluation or the reproducibility of the pipeline. Feature selection and scaling were fitted

before the final split, the nominal test set was used during model selection, categorical labels were coded independently by chunk, and the reported test sample count is inconsistent with the confusion-matrix denominator. In addition, class weights were calculated after SMOTE had already balanced the training classes, making the weights effectively redundant. The rate features were also calculated after global scaling, which weakens their interpretation as physical rates.

Table 4. Publication-relevant methodological issues and required corrections.

Issue	Implication	Required correction
Feature-selection leakage	Test labels can influence selected predictors	Split first; fit selector on training data only
Scaling leakage	Test distribution influences transformation parameters	Fit all learned scalers on training data only
Test used for model selection	Final holdout is not independent	Create a separate validation subset
Chunk-wise label coding	Attack-only chunks may receive code 0	Map original text labels explicitly
Redundant class weighting	Weights add little after SMOTE balancing	Use one justified imbalance strategy
Sample-count inconsistency	Evaluation denominator cannot be reconstructed	Log row and class counts after each stage
Rates after global scaling	Engineered ratios lose physical interpretation	Engineer rate variables before scaling

DISCUSSION

The empirical result supports the feasibility of 1D convolution for binary intrusion screening on flow-level CSE-CIC-IDS2018 data. Accuracy near 95.2% and reported ROC-AUC near 97.5% indicate substantial separation between benign and malicious records in the evaluated partition. The result is lower than values reported in some CSE-CIC-IDS2018 studies, but direct ranking would be misleading because preprocessing, class definitions, dataset subsets, and validation protocols differ. CNN-based IDS results are particularly difficult to compare when experimental procedures are not standardized [4]. The appropriate interpretation is therefore a credible internal benchmark whose external competitiveness depends on a controlled rerun.

The precision-recall profile is more informative than accuracy alone. Recall above 90% indicates that most malicious flows in the evaluated partition were detected. Precision below 86% means that roughly one in seven malicious predictions was a false alert. The false-positive rate derived from the confusion matrix is approximately 3.58%. In a high-volume environment, that percentage can translate into a large absolute alert burden. A deployment-oriented IDS study should therefore report classification metrics together with alert volume or false alarms per unit time under realistic benign traffic. The source experiment does not provide such an operational analysis, so that claim remains outside the evidence available here.

The model design itself is reasonable as a compact research prototype. Conv1D layers with batch normalization and dropout provide nonlinear representation capacity, while mutual-information selection and SMOTE address dimensionality and class imbalance. Yet the methodology demonstrates a recurring problem in machine-learning cybersecurity research: individually reasonable components can be combined in an evaluation order that makes the final estimate optimistic. Supervised feature selection fitted on the full dataset allows information about test labels to affect which features reach the model. Global scaling also permits the future test distribution to influence training representations. Reusing the test set for early stopping then creates a direct path from test labels to model selection. These issues are more than implementation details because leakage can materially change measured performance [9,11].

The source code also raises a label-integrity concern. Converting text labels to categorical codes inside independent chunks assumes that code 0 always means benign. That assumption may hold when benign

records are present and category ordering places Benign first, but it is not guaranteed for an attack-only chunk. The safer method is deterministic binary mapping from the original text label before any per-chunk encoding. Assertions should verify class counts per source file and confirm that no attack label has been mapped to the benign class.

The discrepancy between the approximate test-set size and the confusion-matrix total requires similar restraint. It may reflect row removal, use of a subset of files, a different run, or a reporting error. The source material does not identify the cause. A rerun should therefore create an audit log recording rows loaded, duplicates removed, rows excluded during cleaning, rows retained after feature filtering, class counts before and after SMOTE, and final training, validation, and test sizes. This audit trail is necessary for exact reproducibility.

A broader conceptual limitation concerns applying Conv1D to tabular flow features. Convolution assumes that local neighbourhoods carry meaningful structure. In images, neighbouring pixels are naturally related. In a flow-feature vector, however, column order is chosen by the researcher. A kernel of size 3 therefore learns interactions among adjacent features that may depend on an arbitrary ordering. This does not make Conv1D invalid, but it means feature ordering should be justified or tested. An ablation that permutes feature order, groups semantically related variables, or compares the CNN against a multilayer perceptron and strong tabular baselines would clarify whether convolution provides a genuine advantage.

The strongest future comparisons are models trained on the same corrected split and preprocessing pipeline rather than cross-paper accuracy tables. Logistic regression can provide a linear reference. A tree ensemble can provide a strong non-neural baseline. A multilayer perceptron can test whether convolution adds value beyond dense nonlinear layers. If the 1D-CNN does not outperform these baselines on the same data, its contribution should be framed around computational efficiency, robustness, or integration rather than predictive superiority. Cross-dataset or temporal testing should be added when the intended claim extends beyond internal benchmark discrimination [10].

Methodological limitations and leakage-controlled replication design

A confirmatory replication should preserve the original binary task and core 1D-CNN hypothesis while correcting the order of data-dependent operations. The dataset should be loaded with deterministic file ordering and explicit text-to-binary label mapping. Row counts and class distributions should be logged after every cleaning step. A fixed training, validation, and test partition should then be created before any learned imputation, scaling, feature ranking, or resampling.

Within the training partition, missing-value parameters, scaling parameters, and mutual-information feature selection should be fitted. SMOTE, if retained, should operate only on the training fold. The validation partition should control early stopping, learning-rate reduction, threshold selection, and architecture choices. Once the model is frozen, the untouched test set should produce the final confusion matrix and metrics. At least five random seeds should be reported for random-split evaluation using mean and standard deviation. If reliable timestamps or host identifiers are available, a chronological or grouped split should be added to reduce dependence between partitions. Cross-dataset testing is preferable when publication claims concern deployment-level generalization [10].

Threats to validity

The principal internal-validity threat is that information from the nominal test observations entered model development through global scaling, supervised feature selection, and validation-based checkpoint selection. These paths make the reported test metrics partly conditional on decisions informed by the same partition. Construct validity is affected by two implementation choices. First, chunk-wise categorical coding does not guarantee a stable benign code across all chunks. Second, packet-rate variables calculated after global scaling no longer retain a straightforward interpretation in physical units. Both issues can be corrected without changing the binary classification objective.

External validity remains limited because the selected experiment uses an internal random split from one

benchmark. Related flows generated in the same simulated environment may share capture-specific characteristics across partitions. The present evidence therefore supports within-benchmark discrimination, not cross-environment deployment performance. Chronological, grouped, or cross-dataset evaluation would provide a stronger test of transfer. Reproducibility is also constrained by the discrepancy between the approximate 3.2 million test observations described in the source experiment and the 1,572,022 observations represented by the selected confusion matrix. Exact stage-wise row counts, deterministic split identifiers, software versions, random seeds, and final preprocessing parameters should be archived in a confirmatory rerun. These limitations define the boundary of the current claims rather than nullifying the observed classification result.

CONCLUSION

The evaluated 1D-CNN demonstrates useful within-benchmark discrimination on CSE-CIC-IDS2018, achieving 95.206% accuracy, 96.421% specificity, 85.747% precision, 90.122% recall, and 87.880% F1, with a reported ROC-AUC of 97.47%. These results support feasibility for binary intrusion screening, but they do not establish CNN superiority or deployment-level generalization. No simpler baseline was evaluated under the same pipeline, so the added value of convolution remains unestablished. The absence of ablation studies also prevents attribution of performance to individual components. External validity is limited by one dataset and an internal random split, while statistical robustness remains unresolved because results derive from a selected run without repeated seeds or confidence intervals. Reproducibility is constrained by the absence of released code, explicit seeds, software versions, split identifiers, and complete processing records. These weaknesses compound the identified risks from preprocessing before partitioning, test-set reuse, chunk-wise label coding, and inconsistent sample counts. The contribution is therefore best interpreted as a performance evaluation and methodological audit. A confirmatory replication should use leakage-controlled train-validation-test separation, compare representative linear, tree-based, and non-convolutional neural baselines, perform component-level ablations, repeat experiments across documented seeds with uncertainty estimates, and add chronological, grouped, or cross-dataset testing. Releasing the implementation and environment specifications would further strengthen reproducibility and support stronger claims concerning robustness and generalization.

DECLARATIONS

Data availability. CSE-CIC-IDS2018 is a publicly available benchmark distributed by the Canadian Institute for Cybersecurity. A leakage-controlled rerun should archive the exact preprocessing configuration and final split identifiers used for evaluation.

Ethics. The study analyzes a publicly available network-flow benchmark dataset and does not involve human participants or animals.

Competing interests. To be confirmed by all listed authors before submission.

REFERENCES

1. Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1), e4150. <https://doi.org/10.1002/ett.4150>
2. Chou, D., & Jiang, M. (2021). A survey on data-driven network intrusion detection. *ACM Computing Surveys*, 54(9), Article 182. <https://doi.org/10.1145/3472753>
3. Magan-Carrion, R., Urda, D., Diaz-Cano, I., & Dorronsoro, B. (2020). Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches. *Applied Sciences*, 10(5), 1775. <https://doi.org/10.3390/app10051775>
4. Mohammadpour, L., Ling, T. C., Liew, C. S., & Aryanfar, A. (2022). A survey of CNN-based network intrusion detection. *Applied Sciences*, 12(16), 8162. <https://doi.org/10.3390/app12168162>

5. Qazi, E. H., Almorjan, A., & Zia, T. A. (2022). A one-dimensional convolutional neural network (1D-CNN) based deep learning system for network intrusion detection. *Applied Sciences*, 12, 7986. <https://doi.org/10.3390/app12167986>
6. Wang, Y. C., Houn, Y. C., Chen, H. X., & Tseng, S. (2023). Network anomaly intrusion detection based on deep learning approach. *Sensors*, 23, 2171. <https://doi.org/10.3390/s23042171>
7. Wang, Z., & Ghaleb, F. A. (2023). An attention-based convolutional neural network for intrusion detection model. *IEEE Access*, 11, 43116-43127. <https://doi.org/10.1109/ACCESS.2023.3271408>
8. Anwar, R. W., Abrar, M., Salam, A., & Ullah, F. (2026). A lightweight 1D convolutional neural network for malicious network traffic detection on heterogeneous benchmark datasets. *Array*, 30, 100925. <https://doi.org/10.1016/j.array.2026.100925>
9. Mallampati, S. B., & Hari, S. (2024). A comparative study on the impacts of data leakage during feature selection using the CIC-IoT 2023 intrusion detection dataset. 2024 10th International Conference on Electrical Energy Systems, 1-6. <https://doi.org/10.1109/ICEES61253.2024.10776873>
10. Apruzzese, G., Pajola, L., & Conti, M. (2022). The cross-evaluation of machine learning-based network intrusion detection systems. *IEEE Transactions on Network and Service Management*, 19, 5152-5169. <https://doi.org/10.1109/TNSM.2022.3157344>
11. Aburomman, A., Alakhras, M., & Reaz, M. (2026). Leakage-safe IoT intrusion detection with White Shark Optimizer-based feature selection. *Array*, 30, 100903. <https://doi.org/10.1016/j.array.2026.100903>
12. Kim, A., Park, M., & Lee, D. H. (2020). AI-IDS: Application of deep learning to real-time web intrusion detection. *IEEE Access*, 8, 70245-70261. <https://doi.org/10.1109/ACCESS.2020.2986882>
13. Mo, Y., Li, H., Wang, D., & Liu, G. (2024). An intrusion detection system based on convolution neural network. *PeerJ Computer Science*, 10, e2152. <https://doi.org/10.7717/peerj-cs.2152>
14. Hidayat, I., Ali, M. Z., & Arshad, A. (2022). Machine learning-based intrusion detection system: An experimental comparison. *Journal of Computational and Cognitive Engineering*, 2(2), 88-97. <https://doi.org/10.47852/bonviewJCCE2202270>
15. Hu, Z., Wang, L., Qi, L., Li, Y., & Yang, W. (2020). A novel wireless network intrusion detection method based on adaptive synthetic sampling and an improved convolutional neural network. *IEEE Access*, 8, 195741-195751. <https://doi.org/10.1109/ACCESS.2020.3034015>
16. Vanin, P., Newe, T., Dhirani, L. L., O'Connell, E., O'Shea, D., Lee, B., & Rao, M. (2022). A study of network intrusion detection systems using artificial intelligence/machine learning. *Applied Sciences*, 12(22), 11752. <https://doi.org/10.3390/app122211752>
17. Keserwani, P. K., Govil, M. C., & Pilli, E. S. (2023). An effective NIDS framework based on a comprehensive survey of feature optimization and classification techniques. *Neural Computing and Applications*, 35(7), 4993-5013.
18. Yang, Z., Liu, Z., Zong, X., & Wang, G. (2022). An optimized adaptive ensemble model with feature selection for network intrusion detection. *Concurrency and Computation: Practice and Experience*, 35(4), e7529.