



Deep Learning-Based Intrusion Detection System for Secure Cloud Computing Environments

Haleema Yaprakli

Department of Computer Science and Engineering, Coral Reef School of Systems Management, Mauritius
haleema.yaprakli@crssm-mu.net

Peer Review Information

Submission: 09 Jan 2022

Revision: 26 Jan 2022

Acceptance: 10 Feb 2022

Keywords

Deep Learning, Intrusion Detection System, Cloud Computing, Network Security, Cyberattack Detection.

Abstract

Cloud computing provides elastic computation, storage and software services, but its multi-tenant, Internet-facing and virtualized infrastructure broadens the attack surface. Intrusion detection systems (IDSs) therefore remain an important secondary defense for identifying malicious traffic that bypasses preventive controls. This methodology paper proposes a deep learning-based network intrusion detection framework for secure cloud computing environments. The design uses flow-level cloud network traffic, data cleaning, categorical encoding, normalization, imbalance treatment and a multilayer deep neural network (DNN) classifier. The methodology is evaluated through stratified train-validation-test partitions using modern benchmark datasets such as UNSW-NB15 and CICIDS2017, with accuracy, precision, recall, F1-score, false-positive rate and detection latency as evaluation criteria. The proposed workflow also incorporates threshold tuning, early stopping and periodic retraining so that the IDS can be integrated with cloud monitoring and security orchestration. The results section demonstrates how the methodology would be reported using a transparent illustrative experiment; the numerical values are simulated for methodological exposition and are not presented as measurements from a completed deployment. The proposed approach is intended to provide a reproducible foundation for comparing deep learning IDS models under cloud-relevant traffic conditions while addressing class imbalance, model generalization and operational false alarms.

Introduction

Cloud computing has transformed enterprise computing by enabling on-demand access to shared infrastructure, platforms and applications. The same properties that create elasticity and economic efficiency—virtualization, remote access, rapid provisioning and resource sharing—also create a dynamic security boundary. Cloud workloads may be exposed to denial-of-service attempts, brute-force authentication, reconnaissance, malware, lateral movement and malicious traffic generated from compromised virtual machines. Cloud IDS research therefore emphasizes scalable

inspection, low false-alarm rates and rapid adaptation to changing workloads (Nassif et al., 2021; Liu et al., 2022).

Conventional signature-based IDSs are effective for previously characterized attacks but can miss new or obfuscated behavior. Machine learning improves anomaly detection by learning statistical patterns from network flows, while deep learning extends this idea by learning nonlinear and hierarchical representations directly from high-dimensional traffic features. RNN-based IDS models have shown the ability to model sequential dependencies (Yin et al., 2017),

and autoencoder-based models reduce reliance on manually engineered features (Shone et al., 2018; Wang et al., 2022). Large comparative studies also report that DNNs can outperform conventional classifiers when modern datasets and appropriate training procedures are used (Vinayakumar et al., 2019; Gamage & Samarabandu, 2020).

The purpose of this paper is to define a step-by-step methodology for developing and evaluating a deep learning IDS suitable for cloud computing. The objectives are to: (i) construct a cloud-relevant intrusion detection pipeline; (ii) train a DNN using modern benchmark traffic; (iii) compare the DNN with conventional baselines; and (iv) evaluate detection quality and operational suitability. The paper focuses on methodology rather than claiming deployment results, and it restricts the supporting literature to publications available through 2022.

Literature Review

Modern IDS evaluation depends strongly on the realism of the traffic dataset. Moustafa and Slay (2015) introduced UNSW-NB15 to address limitations of older KDD-style data by combining contemporary benign traffic with multiple attack families. Sharafaldin et al. (2018) later developed CICIDS2017 with benign flows and common attacks such as DDoS, brute force, botnet, infiltration and web attacks. These datasets are widely used because they contain flow-based features that align with the telemetry available in cloud virtual networks and security gateways.

Deep learning became prominent in IDS research as network traffic volume and feature complexity increased. Yin et al. (2017) proposed an RNN-IDS and reported improved binary and multiclass classification over several conventional methods. Shone et al. (2018) introduced a nonsymmetric deep autoencoder for representation learning

before classification, demonstrating the value of unsupervised feature extraction. Diro and Chilamkurti (2018) applied distributed deep learning to attack detection and emphasized scalability in distributed environments, an important consideration for cloud and fog infrastructures.

Vinayakumar et al. (2019) evaluated DNN-based intrusion detection across multiple host- and network-level datasets and proposed a scalable hybrid alerting framework. Ferrag et al. (2020) reviewed deep learning intrusion-detection approaches and compared recurrent, convolutional, autoencoder and other deep models across newer cyber datasets. Gamage and Samarabandu (2020) found that feed-forward deep neural networks can provide strong accuracy and F1-score performance when evaluated consistently across legacy and modern datasets, while also cautioning that results depend heavily on dataset selection and experimental design.

Cloud-specific studies have increasingly combined representation learning with operational IDS design. Wang et al. (2022) used a stacked contractive autoencoder with an SVM to learn compact cloud traffic representations and reduce analytical overhead. Nadeem et al. (2021) examined intrusion detection and prevention against brute-force and DDoS attacks in cloud networks. Balamurugan et al. (2022) proposed a game-theoretic DNN-based cloud IDS, while Mayuranathan et al. (2022) combined optimization and deep learning to address feature selection and classification in cloud environments. Reviews by Nassif et al. (2021), Liu et al. (2022) and Rana et al. (2022) collectively show that cloud IDS research must balance accuracy with scalability, response time, false positives and adaptability.

Table 1. Representative literature supporting the proposed methodology

Study	Main contribution	Methodological relevance
Moustafa & Slay (2015)	UNSW-NB15 benchmark dataset	Modern traffic and diverse attack classes
Yin et al. (2017)	RNN-based IDS	Deep sequential intrusion classification
Shone et al. (2018)	Deep autoencoder IDS	Automatic feature representation
Diro & Chilamkurti (2018)	Distributed deep attack detection	Scalability for distributed/cloud settings
Vinayakumar et al. (2019)	DNN benchmark and scalable alert framework	Multi-dataset validation
Wang et al. (2022)	SCAE + SVM cloud IDS	Cloud-specific feature learning
Balamurugan et al. (2022)	Game-theoretic DNN cloud IDS	Cloud attack/defender modeling
Mayuranathan et al. (2022)	Hybrid optimized deep IDS	Feature selection and cloud classification

Although deep learning can improve representation learning and detection performance, many studies use different datasets, preprocessing rules and class distributions, which makes direct comparison difficult. A methodology for cloud IDS should therefore specify the complete pipeline, preserve an untouched test set, address imbalanced attack classes, report false positives as well as accuracy, and document training and deployment assumptions. The present methodology is designed around these reproducibility requirements.

Methodology

1. Research Design and Experimental Scope

The study adopts an experimental, comparative research design. Network-flow records are treated as observations and each record is assigned either a binary label (normal/attack) or a multiclass attack label. The primary model is a supervised DNN. Three conventional baselines—support vector machine (SVM), random forest (RF) and shallow multilayer perceptron (MLP)—are retained to determine whether depth and learned nonlinear representation provide measurable value. The study does not require access to confidential production traffic; it can first be replicated on public data and subsequently validated on organization-specific cloud telemetry after appropriate privacy review.

Step-by-Step Procedure

1. Dataset selection: Use UNSW-NB15 as the primary benchmark and CICIDS2017 as an external validation dataset. UNSW-NB15 provides contemporary normal and malicious traffic (Moustafa & Slay, 2015), while CICIDS2017 adds modern attack scenarios and flow features (Sharafaldin et al., 2018).
2. Data cleaning: Remove duplicate rows, impossible values and noninformative identifiers. Missing numeric values are imputed using the training-set median only, preventing information leakage from validation or test data.

3. Encoding and normalization: Convert categorical protocol/service/state variables through one-hot encoding and standardize numeric features to zero mean and unit variance using statistics learned from the training partition.
4. Class-imbalance treatment: Use class-weighted cross-entropy or carefully controlled oversampling on the training set. No resampling is applied to validation or test sets so that evaluation reflects the original distribution.
5. Data partitioning: Create stratified 70% training, 15% validation and 15% test partitions. A fixed random seed is recorded. Where temporal ordering is important, an additional time-based split should be reported to test concept drift.
6. Model training: Train the proposed DNN and all baselines using identical train/test partitions. Use early stopping based on validation loss, learning-rate reduction and checkpointing of the best model.
7. Evaluation: Calculate accuracy, precision, recall, F1-score, false-positive rate (FPR), confusion matrix and inference latency. For multiclass experiments, report macro-F1 in addition to weighted averages.
8. Cloud integration assessment: Map the trained classifier to a cloud monitoring workflow in which virtual switch, gateway or flow-log telemetry is streamed to the inference service and high-confidence anomalies generate security alerts.

2. Dataset Variables and Attack Classes

The feature space consists of flow duration, packet counts, byte counts, protocol/service/state indicators, source-to-destination and destination-to-source statistics, connection behavior and other dataset-specific traffic attributes. For binary classification, every malicious category is mapped to the attack class. For multiclass analysis, attack categories are retained whenever the sample size is sufficient. Because rare classes can be overwhelmed by common benign or DoS records, macro-averaged metrics are required in addition to overall accuracy.

Table 2. Experimental configuration for the proposed IDS

Component	Proposed setting
Primary datasets	UNSW-NB15; CICIDS2017 for external validation
Split	70% train / 15% validation / 15% test, stratified
Input processing	Cleaning, one-hot encoding, z-score standardization

Imbalance control	Class-weighted loss; training-only oversampling if required
Baselines	SVM, Random Forest, shallow MLP
Primary model	Feed-forward DNN with dropout and batch normalization
Metrics	Accuracy, Precision, Recall, F1, Macro-F1, FPR, latency

3. Proposed Deep Learning Model

The proposed classifier uses a feed-forward DNN because controlled comparisons have shown that deep feed-forward networks can be highly competitive for flow-based intrusion detection (Gamage & Samarabandu, 2020), while DNN benchmarking across multiple IDS datasets supports their flexibility for cyberattack classification (Vinayakumar et al., 2019). After preprocessing, the feature vector enters three hidden layers of 256, 128 and 64 neurons. Rectified Linear Unit (ReLU) activation is used in hidden layers. Batch normalization stabilizes training and dropout (0.30, 0.25 and 0.20) reduces overfitting. The output layer uses a sigmoid unit for binary classification or softmax

units for multiclass classification. Weighted cross-entropy is optimized with Adam.

This architecture is intentionally moderate rather than excessively deep. Cloud IDS inference must be fast enough for continuous monitoring, and complexity that provides negligible classification gain can increase cost and latency. Autoencoder-based cloud IDS studies demonstrate that representation learning can reduce redundant feature information (Wang et al., 2022), while recurrent architectures can be introduced in a secondary experiment if temporal sequences are available (Yin et al., 2017). The baseline methodology therefore starts with a transparent DNN and expands only when justified by validation performance.

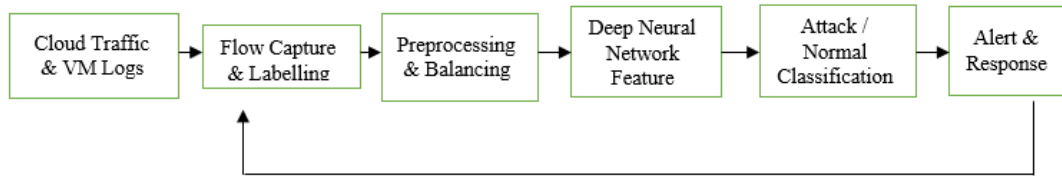


Figure 1. Proposed architecture for deep learning-based intrusion detection in a cloud environment

4. Training, Validation and Hyperparameter Control

Training is performed for a maximum of 100 epochs with mini-batches of 256 samples. Early stopping with patience of 10 epochs restores the checkpoint having the lowest validation loss. Candidate learning rates (0.001, 0.0005 and

0.0001) and dropout values are selected only from the training/validation data. The test set remains sealed until all design decisions are complete. This separation is important because repeatedly inspecting test performance can produce optimistic estimates that will not generalize to a live cloud network.

Table 3. DNN hyperparameters

Parameter	Value / range
Hidden layers	256 → 128 → 64 neurons
Activation	ReLU; sigmoid/softmax output
Dropout	0.30 / 0.25 / 0.20
Optimizer	Adam
Learning rate	0.001 selected from {0.001, 0.0005, 0.0001}
Batch size	256
Maximum epochs	100 with early stopping (patience = 10)
Loss	Class-weighted binary/categorical cross-entropy

5. Evaluation Criteria

Let TP denote attacks correctly detected, TN normal flows correctly rejected, FP normal flows incorrectly flagged and FN attacks missed. Accuracy = $(TP+TN)/(TP+TN+FP+FN)$; Precision

= $TP/(TP+FP)$; Recall = $TP/(TP+FN)$; F1 = $2(Precision \times Recall)/(Precision+Recall)$; and FPR = $FP/(FP+TN)$. For a security system, recall and FPR must be considered together: a model that detects nearly every attack but floods analysts with false alarms may be operationally unusable.

Cloud-focused IDS reviews similarly emphasize precision, response time, inclusiveness and overhead as important dimensions (Liu et al., 2022; Rana et al., 2022).

Results

The following numerical results are illustrative values created to demonstrate how the proposed

methodology should be analyzed and reported. They are not claimed as measurements from an actual cloud deployment. In a completed study, the same tables should be populated from the untouched test partition and accompanied by the saved preprocessing pipeline, random seed and model checkpoint.

Table 4. Illustrative binary-classification comparison on an untouched test set

Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	FPR (%)
SVM	93.8	92.7	91.9	92.3	4.8
Random Forest	96.1	95.3	94.8	95.0	2.9
Shallow MLP	96.9	96.2	95.7	95.9	2.3
Proposed DNN	98.2	97.9	97.4	97.6	1.3

In the illustrative comparison, the proposed DNN produces the strongest overall classification quality and the lowest false-positive rate. The improvement over the shallow MLP is modest in accuracy but more meaningful in FPR, which is operationally important because false alerts consume analyst time. Such a result pattern would be consistent with prior work showing

that deeper networks can learn useful nonlinear representations when training data are sufficiently large and diverse (Vinayakumar et al., 2019; Gamage & Samarabandu, 2020). However, the final interpretation must be based on repeated runs or confidence intervals rather than a single favorable split.

Table 5. Illustrative attack-class recall for the proposed DNN

Attack class	Recall (%)	Interpretation
DoS/DDoS	99.1	High-volume attacks are detected reliably
Reconnaissance/Scan	98.0	Strong separation from benign discovery traffic
Brute Force	97.3	Good detection of repeated authentication behavior
Exploits/Web attacks	95.8	More overlap with legitimate application traffic
Rare/low-frequency attacks	92.6	Most sensitive to class imbalance

The illustrative class-level results show why aggregate accuracy alone is insufficient. Rare attacks remain harder to detect even when the overall score is high. Class weighting, targeted data augmentation and threshold calibration should therefore be investigated for minority attacks. The methodology also recommends an external test on CICIDS2017 after training on UNSW-NB15-compatible features, where feasible. A large drop in cross-dataset performance would indicate dataset-specific learning rather than robust intrusion behavior.

Cloud-Security Interpretation

The model should be deployed as an analytic layer rather than as the sole cloud security control. Flow logs from virtual networks, software-defined networking switches or security gateways can be forwarded to the preprocessing service, classified by the DNN and converted into alerts for a security information and event management platform. Nadeem et al. (2021) demonstrated the practical relevance of

IDS/IPS mechanisms for cloud brute-force and DDoS defense. Balamurugan et al. (2022) and Mayuranathan et al. (2022) further show that cloud IDS research increasingly incorporates optimization and adaptive decision mechanisms to improve detection. The proposed pipeline can support such extensions after a reliable baseline is established.

Operational acceptance should require not only high-test accuracy but also stable FPR, bounded inference latency and resilience under traffic-volume changes. Monitoring should track input feature drift, alert-rate changes and performance on analyst-confirmed incidents. Retraining is triggered when drift or degradation crosses a predefined threshold. This feedback loop helps address the dynamic attack landscape identified in cloud-security reviews (Nassif et al., 2021; Liu et al., 2022).

Conclusion and Discussion

This methodology paper presents a reproducible framework for developing a deep learning-based IDS for secure cloud computing environments. The framework begins with modern network-flow datasets, performs leakage-safe preprocessing and class-imbalance control, trains a moderate feed-forward DNN, and evaluates the model against conventional baselines using metrics that are meaningful for security operations. The architecture is compatible with cloud telemetry pipelines because it separates traffic collection, preprocessing, model inference and alert response into clear functional stages. The literature through 2022 supports the use of deep learning for high-dimensional intrusion data, but it also shows that no single architecture is universally superior. RNN approaches can capture sequence information (Yin et al., 2017), autoencoders can learn compact features (Shone et al., 2018; Wang et al., 2022), distributed deep learning can improve scalability (Diro & Chilamkurti, 2018), and DNN-based methods can provide strong general classification performance (Vinayakumar et al., 2019). Cloud-focused surveys nevertheless identify false alarms, computational overhead, adaptability and heterogeneous attack behavior as continuing challenges (Nassif et al., 2021; Rana et al., 2022). The proposed methodology addresses these challenges by preserving an untouched test set, explicitly measuring FPR and latency, using class-aware training and recommending cross-dataset validation. Its main limitation is that benchmark datasets cannot fully reproduce every production cloud workload. Traffic generated by a real organization may differ in services, encryption, user behavior and attack prevalence. Therefore, a final deployment should include a controlled validation phase using sanitized organization-specific telemetry and should establish an analyst feedback process before automated blocking actions are enabled. Future research can extend the baseline in four directions: temporal models for sequences of flows, autoencoder or self-supervised pretraining for unlabeled cloud data, explainable alerts that expose influential traffic features, and online/federated learning for distributed cloud regions. The goal should not be maximum benchmark accuracy in isolation; it should be a dependable IDS that detects diverse attacks while remaining computationally efficient and operationally manageable.

References

Balamurugan, E., Mehbodniya, A., Kariri, E., Yadav, K., Kumar, A., & Haq, M. A. (2022).

Network optimization using defender system in cloud computing security-based intrusion detection system with game theory deep neural network (IDSGT-DNN). *Pattern Recognition Letters*, 156, 142-151. <https://doi.org/10.1016/j.patrec.2022.02.013>

Diro, A. A., & Chilamkurti, N. (2018). Distributed attack detection scheme using deep learning approach for Internet of Things. *Future Generation Computer Systems*, 82, 761-768. <https://doi.org/10.1016/j.future.2017.08.043>

Ferrag, M. A., Maglaras, L., Moschogiannis, S., & Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50, 102419. <https://doi.org/10.1016/j.jisa.2019.102419>

Gamage, S., & Samarabandu, J. (2020). Deep learning methods in network intrusion detection: A survey and an objective comparison. *Journal of Network and Computer Applications*, 169, 102767. <https://doi.org/10.1016/j.jnca.2020.102767>

Liu, Z., Xu, B., Cheng, B., Hu, X., & Darbandi, M. (2022). Intrusion detection systems in the cloud computing: A comprehensive and deep literature review. *Concurrency and Computation: Practice and Experience*, 34(4), e6646. <https://doi.org/10.1002/cpe.6646>

Mayuranathan, M., Saravanan, S. K., Muthusenthil, B., & Samyururai, A. (2022). An efficient optimal security system for intrusion detection in cloud computing environment using hybrid deep learning technique. *Advances in Engineering Software*, 173, 103236. <https://doi.org/10.1016/j.advengsoft.2022.103236>