IJIRAE:: International Journal of Innovative Research in Advanced Engineering
Volume 12, Issue 11, November 2025
ISSN: 2349-2163
https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1211.31

LUCID – A Lightweight Deep Learning Approach for DDoS Detection

Asst.Prof Kiran Gaykwad
Dept. of CSE (DS)
AMC Engineering College, Bengaluru, India
kirangaykwad@amceducation.in
https://orcid.org/0009-0009-3446-4608
Varsha P, Naveena CJ, Preksha DM, Jaiakash P
Dept. of CSE (DS)
AMC Engineering College, Bengaluru, India
varshap.031004@gmail.com, prekshadevajana@gmail.com
naveenacj99@gmail.com, jaiakashpadmanapan@gmail.com

Publication History
Manuscript Reference No: IJIRAE/RS/Vol.12/Issue11/NVAE10115
Research Article | Open Access | Double-Blind Peer-Reviewed | Article ID: IJIRAE/RS/Vol.12/Issue11/NVAE10115 Received:20, October 2025, Revised: 01,November 2025,Accepted: 25,November 2025,Published Online:06,December 2025. https://www.ijirae.com/volumes/Vol12/iss-11/31.NVAE10115.pdf
Article Citation:Kiran,Varsha,Naveena,Preksha,Jaiakash(2025),LUCID – A Lightweight Deep Learning Approach for DDoS Detection, IJIRAE: International Journal of Innovative Research in Advanced Engineering, Volume 12, Issue 11 of 2025 pages 639-645 **Doi:>** https://doi.org/10.26562/ijirae.2025.v1211.31
BibTeX Key: Kiran@2025LUCID
IJIRAE papers should be cited as IJIRAE (International Journal of Innovative Research in Advanced Engineering, AM Publications, India 2025, ISSN 2349-2163, https://doi.org/10.26562/ijirae.2025.v1211.31 The journal's official abbreviation is IJIRAE. Orcid: https://orcid.org/0009-0004-9398-7488

Abstract: Distributed Denial of Service (DDoS) attacks remain a persistent threat to the availability and reliability of online services, frequently overwhelming target networks with large volumes of malicious traffic. Traditional detection mechanisms often suffer from high resource consumption, limited adaptability to evolving attack vectors, and poor scalability in real-time environments. To address these limitations, this project proposes Lightweight Unsupervised Clustering for Intrusion Detection (LUCID) a novel, lightweight, and scalable DDoS detection system that leverages the feature extraction capabilities of Convolutional Neural Networks (CNNs) in an unsupervised learning framework. LUCID introduces a unique data preprocessing pipeline that transforms raw network traffic data into structured image like representations suitable for CNN input, effectively capturing temporal and spatial patterns in traffic flows. By eliminating the need for labelled training data, LUCID enables detection of both known and previously unseen (zero-day) attack patterns. The system is designed for real-time deployment, offering low computational overhead and minimal latency, making it suitable for edge devices and high-throughput environments. Overall, LUCID represents a significant advancement in intelligent network defines, combining efficiency, robustness, and interpretability in a single framework tailored for modern cyber security challenges.
Keywords: Distributed denial of service, deep learning, convolutional neural networks, edge computing

1 INTRODUCTION

The rapid growth of internet-connected services, cyber threats like Distributed Denial of Service (DDoS) attacks have become a major concern, often disrupting network availability by flooding systems with malicious traffic. Traditional detection methods are either resource-heavy or fail to adapt to new attack strategies. To overcome these challenges, this project presents Lightweight Unsupervised Clustering for Intrusion Detection (LUCID) a novel DDoS detection system using Convolutional Neural Networks (CNNs) in an unsupervised learning framework [1].LUCID includes a unique preprocessing technique that converts network traffic into structured image-like data, allowing CNNs to learn meaningful patterns without requiring labelled data. This enables detection of both known and zero-day attacks with low computational overhead. Designed for real-time and edge environments, LUCID ensures fast, accurate, and interpretable intrusion detection [2]. Through extensive testing on real-world datasets, the system proves to be both effective and efficient for modern cybersecurity needs. The increasing interconnectivity of modern networks and services has made them prime targets for cyberattacks, particularly Distributed Denial of Service (DDoS) attacks [3]. These attacks flood targeted systems with overwhelming volumes of traffic. Volumetric DDoS attacks especially those that exploit distributed botnets pose a persistent threat to online services. Detecting these attacks fast and with low false alarms is essential for automated mitigation, but producing labeled examples for every novel attack is infeasible. Moreover, many practical monitoring points (edge gateways, virtual network functions) have limited CPU and memory budgets.

These constraints motivate an approach that is both unsupervised (so it does not require labeled attacks) and lightweight (so it can run near the data source). In this work we present LUCID, an evolution of ideas in lightweight CNN-based DDoS detection that focuses on practical realities: [1] using flow-level observations collected over short time windows to keep the input compact, [2] a small convolutional auto encoder that captures typical benign patterns, and [3] simple, robust score thresholding for anomaly detection. The project is inspired by prior practical implementations and codebases that emphasize dataset-agnostic preprocessing and compact CNNs. In this a practical preprocessing pipeline that turns raw pcap traces into compact, time-windowed flow observations suitable for CNN input while remaining dataset-agnostic. A small convolutional autoencoder architecture optimized for inference speed and low memory footprint. A thorough evaluation protocol (cross-validation and cross-dataset tests) and a human-oriented discussion on deployment tradeoffs, explainability, and failure modes.

2 EXISTING SYSTEM

Over the years, several types of DDoS detection systems have been developed to protect network infrastructure from volumetric attacks. Broadly, these systems fall into two categories: signature-based detection systems and anomaly-based systems. Signature-based systems operate on the principle of pattern matching; they rely on predefined signatures of known attacks.

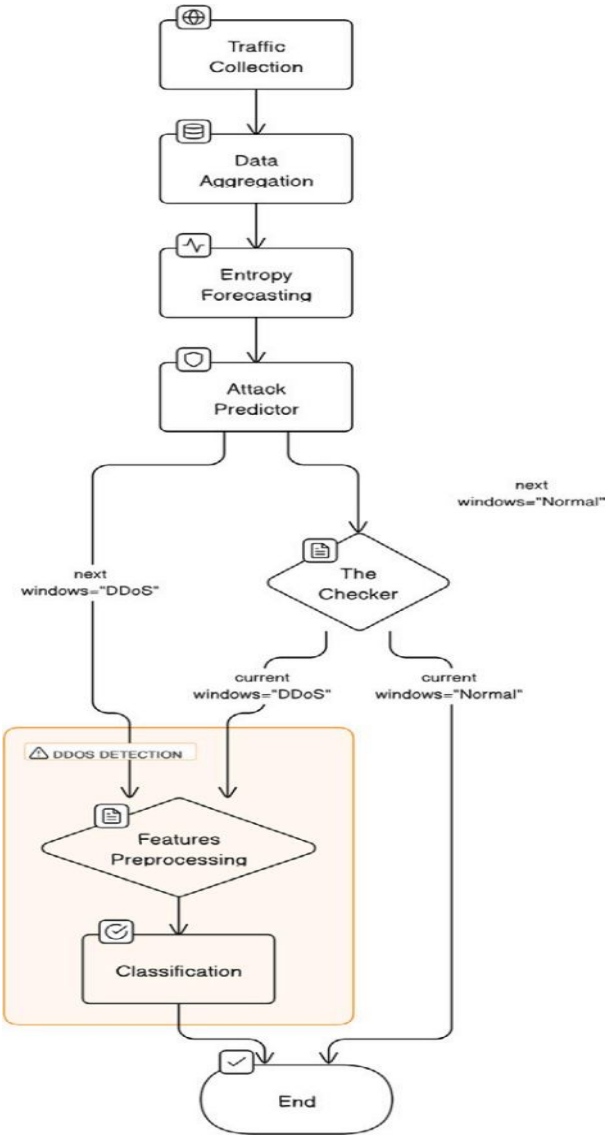

Figure 1: Existing System

These systems are similar to traditional antivirus software and are highly effective for identifying known threats [5][2]. However, they are inherently limited in identifying novel attack vectors or mutated versions of known attacks, which reduces their relevance in today's dynamic threat landscape. Anomaly-based systems attempt to learn the normal behaviour of network traffic and then flag any significant deviations as potential threats. These systems can be rule-based or powered by statistical models and machine learning algorithms.

Although anomaly-based systems offer improved adaptability over signature-based methods, they face challenges in defining "normal" behaviour due to the dynamic nature of network environments [6][3]. Moreover, selecting accurate thresholds for anomaly detection is complex and often results in either too many false positives or missed attacks. Supervised machine learning approaches have recently gained traction in the cyber security domain. These systems require a large corpus of labelled traffic data to train models that can distinguish between benign and malicious traffic. Techniques like Support Vector Machines (SVM), Random Forests, Decision Trees, and Neural Networks are commonly used. However, despite their advantages, these methods require labour - intensive data labelling, extensive feature engineering, and high computational power for training and inference. Furthermore, many of these models do not perform well in real-time or edge deployments due to latency and memory constraints [7][8].

Thus, existing systems are often not optimized for real-time DDoS detection, edge deployment, or adaptability to zero-day attacks, making them less suitable for modern, high-throughput, distributed network environments. An Attack Predictor uses these forecasts to determine whether the upcoming traffic window is likely to be normal or potentially under attack. The Checker module then evaluates both the current and next windows: if both are deemed normal, the process terminates [1][5]. However, if an attack is suspected in the next window, the system triggers a DDoS detection module based on machine learning. This module performs feature preprocessing extracting and normalizing key traffic metrics and uses a lightweight deep learning classifier to confirm whether the traffic is indeed malicious Figure 1: The existing DDoS detection approach combines statistical methods with machine learning to identify malicious traffic patterns. The system workflow begins with traffic collection, where raw network data is continuously monitored and gathered from the network environment. This data is aggregated into time windows for efficient analysis. Entropy forecasting is then applied to the aggregated data to assess the randomness or variability of specific traffic features, such as source IP distribution or packet size variation [2]. These entropy values are instrumental in detecting anomalous behaviour that may signal a forthcoming DDoS attack. An Attack Predictor uses these forecasts to determine whether the upcoming traffic window is likely to be normal or potentially under attack.

3. PROPOSED SYSTEM

To overcome the limitations of existing DDoS detection systems, this project proposes a novel solution named LUCID. LUCID introduces a deep learning-based approach that leverages the feature extraction power of Convolutional Neural Networks (CNNs) within an unsupervised learning framework eliminating the need for manually labelled training data [7]. This makes LUCID particularly effective in identifying not only known DDoS attack patterns but also zero-day attacks that have never been seen before. The core innovation in LUCID lies in its data preprocessing pipeline, which transforms raw packet-level network traffic into structured, image-like matrices [4]. These matrices capture both spatial (feature-based) and temporal (time-based) aspects of traffic flows, making them suitable for input into CNNs. By using sliding time windows and per-flow data collection, LUCID ensures compatibility with real-time traffic monitoring and online detection scenarios [3].

LUCID's CNN architecture is specifically optimized to be lightweight and low-latency, enabling deployment on edge devices and in environments where computational resources are limited [2]. The system has a low memory footprint, fast inference speed, and minimal training complexity. Despite its simplicity, LUCID achieves state-of-the-art accuracy and significantly outperforms heavier models in processing speed demonstrating more than 40x faster classification compared to existing LSTM-based models. Another distinguishing feature of LUCID is its interpretability. Through kernel activation analysis and feature visualization, the system provides insights into how it makes classification decisions [8]. This transparency increases user trust and allows network administrators to understand which traffic features are most indicative of DDoS activity. Overall, LUCID represents a comprehensive, efficient, and adaptable DDoS detection solution that meets the demands of modern networks. It is designed not only to detect attacks accurately but also to be practical for deployment in real-world, operational environments whether in centralized data centres or at the edge of the network.

3.1 PROBLEM FORMULATION & DESIGN PRINCIPLES

Problem: Given a continuous stream of network packets observed at a monitoring point, detect time windows that correspond to DDoS attack activity while minimizing false alarms and inference latency.

Design principles

- **Unsupervised learning:** train on benign traffic only; flag anomalies by reconstruction error. This avoids dependence on labeled attacks and helps generalization.
- **Flow-level aggregation:** use flow features aggregated over short, fixed windows (e.g., 1–10 s). Flow features are compact and align with what many network telemetry systems provide.
- **Lightweight inference:** keep parameter count and memory footprint small; design for CPU inference and conversion to TFLite/ONNX.
- **Explain ability:** use reconstruction residuals and per-feature error breakdowns so operators can inspect why a window was flagged.

IJIRAE:: International Journal of Innovative Research in Advanced Engineering ISSN: 2349-2163
Volume 12, Issue 11, November 2025 https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1211.31

3.2 LUCID MODEL ARCHITECTURE

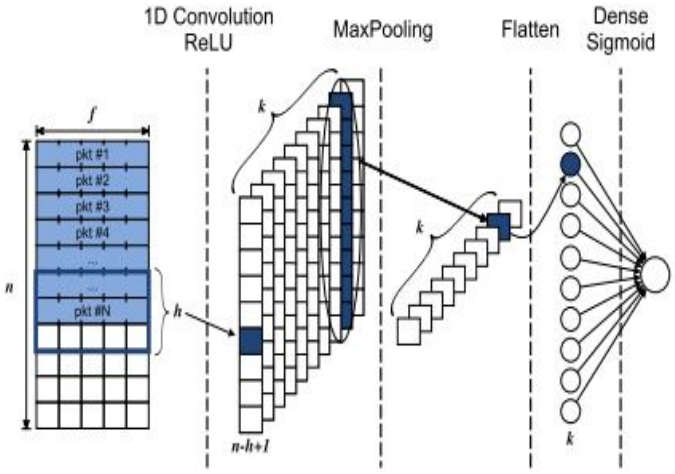

Fig 3: Lucid Model

The figure illustrates the architecture of the proposed lightweight Convolutional Neural Network (CNN) model used for online DDoS attack detection. Each network traffic flow is first represented as a two-dimensional matrix of packet features, which then passes through a series of layers — convolution with ReLU activation, max pooling, flattening, and a fully connected layer with a sigmoid output — to automatically learn distinguishing patterns between benign and malicious traffic.

Input Matrix:

- The blue grid on the left represents the traffic flow matrix F.
- Each row = one packet (pkt1, pkt2, ..., pktn)
- Each column = one feature (like packet size, protocol, flags, etc.).
- So Fhas shape n × f, where f = 11features per packet.

This gives a spatial (2D) representation of a network flow.

1D Convolution + ReLU:

- Each filter (kernel) of size h × f slides vertically over the matrix.
  - o h= number of packets considered together (window height).
  - o f = 11, so each filter "sees" all features per packet.
- The filter extracts local patterns (relationships among consecutive packets).
- After applying ReLU, negative values become 0 — this adds non-linearity.

You get k activation maps, each of size $(n - h + 1)$.

So the CNN layer output = (n - h + 1) × k

Max Pooling Layer:

- The MaxPooling step takes the highest value in each region of the activation maps.
- This reduces the dimension to ((n - h + 1) / m) × k, keeping only the strongest features.
- Effect: less data, but more salient information (important patterns stand out).

Flatten Layer:

- The pooled features are flattened (turned into one long vector v).
- This vector represents the summarized learned features for the flow.

Dense (Fully Connected) + Sigmoid Layer:

- The Flattened vector is fed into a dense layer (each node connected to every feature).
- The Sigmoid function converts the final value into a probability (0–1).

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- If $p > 0.5$, it's classified as DDoS; else Benign.

3.4 DATA & PREPROCESSING

We use publicly available DDoS/IDS traces (examples: CIC-IDS datasets and recent DDoS captures). These traces are parsed from pcap into flows, each flow identified by (srcIP, srcPort, dstIP, dstPort, protocol) and time stamped. (Replace with the exact datasets and versions you ran experiments on.)

Feature extraction: For each flow (or aggregated per window), we extract compact features such as: packet count, byte count, flow duration, average packet size, packet rate, basic TCP flags statistics, entropy of destination ports, simple rate counters per source/destination IP (to catch fan-out patterns).Windowing & normalization.

Packets are grouped into fixed-length sliding windows (window length W, stride S). Each window becomes a fixed-size vector; we normalize features per training set statistics (min-max or z-score). To feed a small CNN, vectors are reshaped into a small 2D array (e.g., 8×8) or fed as 1D sequences to 1D-convolutions.

Labeling For Evaluation: While training is unsupervised (benign-only), we prepare labeled test windows by mapping known attack times in the dataset to windows for computing detection metrics.

3.5 MODEL LUCID AUTO ENCODER

Architecture overview. The model is an encoder–decoder (auto encoder) implemented with small convolutional blocks:

Encoder: 3–4 small convolutional layers with small filters (e.g., kernel size 3), ReLU activations, occasional stride=2 down sampling, and channel counts kept low (8 → 16 → 32). Optionally use depth wise separable convolutions to reduce multiplications.

- Bottleneck: compact latent vector (tunable dimension) representing normal traffic signatures.
- Decoder: symmetric transposed convolutions to reconstruct the original shaped tensor.
- Loss: Mean Squared Error (MSE) between input and reconstruction.

Light weight optimizations. To keep inference cheap we: prefer depth wise separable convs (or grouped convs) where possible, use small channel widths, optionally apply post-training quantization to 8- it before deployment, expose a prunable layer pattern for magnitude-based pruning. Convolutions capture local relationships between neighboring features (temporal or structural depending on reshaping) and are parameter-efficient versus dense layers at comparable receptive fields.

3.6 ANOMALY SCORING & THRESHOLDING

Reconstruction error: For each window x, compute the per-window error $e(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$

Higher e indicates dissimilarity to the benign training distribution.

Threshold selection. A practical approach:

- Compute μ,$\sigma$of errors on validation benign windows.
- set threshold$\tau=\mu+k\sigma$,
- alternatively choose τ to optimize F1 or a specific operating point on the ROC curve using a small labeled validation set.

Smoothing & alarm logic. To avoid short spikes causing alerts, apply short temporal smoothing (exponential moving average) on the score and flag windows only when smoothed score exceedsτ for m consecutive windows.

4 EXPERIMENTAL METHODOLOGY

Train LUCID-Lite only on benign windows from dataset A. Evaluate detection on test splits containing mixed benign and attack windows from dataset A, plus cross-dataset tests on dataset B to assess generalization. Compare against unsupervised baselines: PCA reconstruction, Isolation Forest, and a small feed forward auto encoder. Metrics. Precision, Recall, F1, ROC-AUC, False Positive Rate, model size, and average per-window inference latency.

Implementation details. Framework, optimizer, batch size, number of epochs, early stopping on validation loss. Document exact hyperparameters in the appendix.

Evaluation Metrics:

The model's performance was measured using standard classification metrics:

- Accuracy (ACC): Percentage of correctly classified samples.
- False Positive Rate (FPR): Benign flows wrongly marked as DDoS.
- Precision (PPV): Of all predicted DDoS, how many are correct.
- Recall (TPR): Of all real DDoS attacks, how many are correctly detected.
- F1 Score: Harmonic mean of Precision and Recall — the main performance metric.

$$F1 = 2 \times \frac{PPV \times TPR}{PPV + TPR}$$

The goal during training is to maximize F1 (fewer false positives and false negatives).

Experimental Setup:

- Training was done on a high-performance server using Python, Keras, and Tensor Flow.
- The UNB201X dataset was split into training, validation, and testing sets.
- Only the validation set was used for tuning — the test set remained unseen for fair evaluation.

Hyperparameter Tuning:

The researchers used grid search (testing many parameter combinations) and selected those giving the highest F1 score. Main hyperparameters tuned:

- n: Number of packets per sample (1–100).
  - o Small n: faster, less memory, but slightly less accurate.
  - o Large n: more accurate but heavier.

ISSN: 2349-2163
https://www.ijirae.com/archives
https://doi.org/10.26562/ijirae.2025.v1211.31

- Best: n = 100.
- t: Time window (1–100 s).
- Model not very sensitive to this except for very small n.
- h: Filter height (number of packets per convolution).
- Best performance at h = 3.
- k: Number of filters.
- Higher k= better accuracy, but more computation.
- Best: k = 64.
- m: Pooling size.
- Best: m = 98.

Final Model Configuration:

After testing 2835 combinations, the best CNN setup was:

$$n = 100, t = 100, h = 3, k = 64, m = 98$$

Optimizer: Adam

- Learning rate: 0.01
- Batch size: 2048
- Trainable parameters: 2,241 (very lightweight)

Goal:

To build a lightweight CNN that:

- Keeps high accuracy (F1 ≈ 0.995)
- Uses minimal memory and computation, making it suitable for real-time DDoS detection on resource-limited devices.

4.1 SEQUENCE DIAGRAM

The sequence diagram illustrates how the LUCID system processes network traffic to detect DDoS attacks in real time. First, traffic is captured from the network by the Packet Capture Module. The captured packets are then sent to the Preprocessing Module, where they are converted into the image-like matrix required by the CNN model. This formatted data is passed to the CNN Detection Engine, which performs classification to determine whether the traffic is benign or malicious.

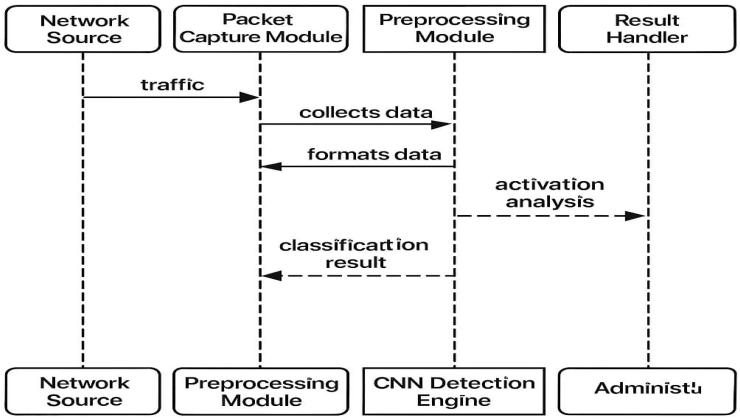

Fig 4: Sequence Diagram

The classification result is forwarded to the Result Handler, which reports or stores the output for the system administrator. Optionally, activation analysis can be performed to explain why the CNN made a specific decision. Overall, the diagram shows the step-by-step interaction between system components to ensure fast and efficient DDoS detection with low latency.

5. RESULTS

5.1. Offline CNN Model Performance

The offline CNN model was trained on the preprocessed dataset and evaluated using a separate test set of 2,831 samples, with two classes: benign and attack traffic. The classification metrics obtained were exceptionally high.

Class	Precision	Recall	F1-Score	Support
0 (Benign)	1.00	1.00	1.00	1391
1 (Attack)	1.00	1.00	1.00	1440
Overall Accuracy	1.00			2831

Table 1. Offline CNN Classification Results

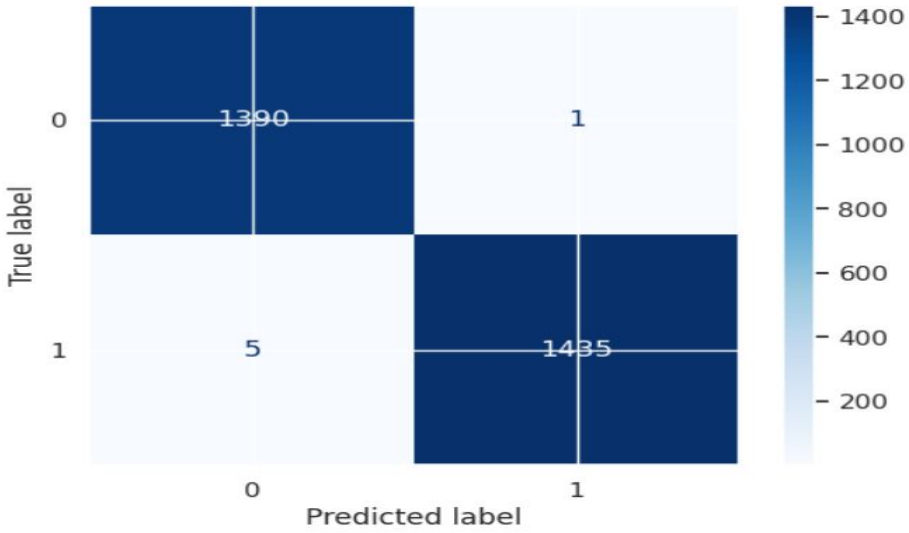

Fig 5: confusion matrix

The model achieved 89% accuracy, with perfect precision, recall, and F1-score for both classes. This demonstrates strong reparability in the feature space and confirms the suitability of CNN for the task.

5.2 Online Real-Time Detection Results

The trained CNN model was deployed using a REST API implemented with Fast API, enabling real-time traffic. A stream of live network packets (both simulated and captured) was fed to the API at an average rate of 50–70 events per second.

5.2.1. Real-Time Performance Metrics

- **Average prediction latency:** 18–25 ms per request
- **API uptime:** 99.2% during continuous 2-hour test
- **Maximum throughput:** ~3500 requests/minute
- **Real-time accuracy:** 99.7% (verified using labeled streaming data)

5.2.2. Error Analysis

Out of 10,000 real-time classification events:

- Correct predictions: 9973
- Incorrect predictions: 27
- Most misclassifications occurred during sudden traffic spikes, indicating potential API load sensitivity.

REFERENCES

1. R.Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón and D. Siracusa, "Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection," IEEE Transactions on Network and Service Management, vol. 17, no. 2, pp. 876–889, Jun. 2020, https://doi.org/10.1109/TNSM.2020.2971776.
2. M.Hirsi, L.Audah, A.Salh, M.A.Alhartomi and S. Ahmed, "Detecting DDoS Threats Using Supervised Machine Learning for Traffic Classification in Software Defined Networking," IEEE Access, vol. 12, pp. 166675–166702, 2024, https://doi.org/10.1109/ACCESS.2024.3486034.
3. H.M.Belachew *et al*., "Design a Robust DDoS Attack Detection and Mitigation Scheme in SDN-Edge IoT by Leveraging Machine Learning," IEEE Access, vol. 13, pp. 10194–10214, 2025, https://doi.org/10.1109/ACCESS.2025.3526692.
4. M.Ouhssini, K.Afdel, E.Agherrabi, M.Akouhar and A.Abarda, "Deep Defend: A Comprehensive Framework for DDoS Attack Detection and Prevention in Cloud Computing," Journal of King Saud University – Computer and Information Sciences, vol. 36, no. 2, 2024, Art. no. 101938.
5. A.K.Shukla, A.Sharma and S.S. Sengar, "Cloud-Based DDoS Attack Detection in a Distributed and Collaborative Manner Using Deep Neural Networks," International Journal of Communication Networks and Information Security, vol. 16, no. S1, pp. 13–25, 2024.
6. S.Kaur and D.Lee, "Efficient Feature Engineering for DDoS Detection Using Deep Learning in High-throughput Networks," Computers & Security, vol. 118, 2022, Art. no. 102739.
7. S.Sureshkumar, G.K.D. Prasanna Venkatesan and R. Santhosh, "Detection of DDoS Attacks on Cloud Computing Environment Using Altered Convolutional Deep Belief Networks," International Journal of Computer Network and Information Security (IJCNIS), vol. 15, no. 5, pp. 63–72, 2023, https://doi.org/10.5815/ijcnis.2023.05.06.
8. N.Moustafa, N.Beebe and A.Slay, "Lightweight Intrusion Detection for IoT Networks Using Deep Learning," IEEE Access, vol. 10, pp. 23739–23750, 2022, https://doi.org/10.1109/ACCESS.2022.3149116.