Unsupervised Deep Learning-Based Network Traffic Anomaly Detection for DDoS Mitigation in Smart Microgrid Communication Infrastructure

Behar Haxhismajli *, Galia Marinova *, Edmond Hajrizi, Besnik Qehaja

Posted Date: 5 May 2026

doi: 10.20944/preprints202605.0082.v1

Keywords: microgrid cybersecurity; DDoS detection; unsupervised anomaly detection; CNN-LSTM; OT protocols; Modbus TCP; MQTT; DNP3; network traffic analysis; Isolation Forest

 doi:10.20944/preprints202605.0082.v1

Article

Unsupervised Deep Learning-Based Network Traffic Anomaly Detection for DDoS Mitigation in Smart Microgrid Communication Infrastructure

Behar Haxhismajli *, Galia Marinova *, Edmond Hajrizi and Besnik Qehaja

Faculty of Telecommunications, Technical University of Sofia, Sofia, Bulgaria

* Correspondence: behar.haxhismajli@ubt-uni.net (B.H.); gim@tu-sofia.bg (G.M.)

Abstract

Smart microgrids depend on continuous communication between controllers, sensors, and actuators over industrial protocols like Modbus TCP, MQTT, and DNP3, that were designed without built-in security mechanisms. The gateway that aggregates this traffic represents a single point of failure vulnerable to distributed denial-of-service (DDoS) attacks. Most existing detection methods require labeled attack data for training, a condition rarely met in operational OT environments. This paper presents an unsupervised CNN-LSTM model trained exclusively on normal microgrid gateway traffic to predict the next traffic window; anomalies are flagged when prediction error exceeds a threshold derived from the training distribution. A dual-branch architecture processes metric time-series through LSTM layers and flow aggregate features through CNN layers, fusing both representations for prediction. The model is evaluated against three protocol-specific DDoS attack scenarios, Modbus SCADA flooding, MQTT publish storm, and DNP3 response flooding - none of which are seen during training. Compared against an Isolation Forest baseline under identical unsupervised conditions, the CNN-LSTM achieves higher precision and recall on all attack types. The framework is deployed within a web-based monitoring platform that supports re-al-time detection and anomaly logging.

Keywords: microgrid cybersecurity; DDoS detection; unsupervised anomaly detection; CNN-LSTM; OT protocols; Modbus TCP; MQTT; DNP3; network traffic analysis; Isolation Forest

1. Introduction

Microgrids are localized energy systems that integrate distributed energy resources-solar inverters, battery storage, wind turbines-with loads and controllers into a coordinated network. These systems rely on continuous data exchange between field devices and a central controller through shared communication infrastructure. The protocols carrying this data are Modbus TCP, MQTT, and DNP3-were designed for reliability in isolated industrial networks, not for security in IP-connected environments. Modbus TCP transmits register values in plaintext over port 502. MQTT brokers in many deployment accept connections with minimal credential verification in the other side DNP3 outstations responds to any master that sends an correctly formatted request. The microgrid gateway, which aggregate all device communication onto a single network interface, becomes a natural target for distributed denial-of-service (DDoS) attack. An sustained flood directed at the gateway, saturates the communication channel, silences sensor telemetry, and disrupt realtime control which potentially causing cascading failures across connected loads [1–4].

A growing of paper researches address intrusion detection for smart grids and a SCADA networks, but the dominants approaches rely most of the time on supervised learning with labeled attack data. Classifiers trained on datasets such as of CIC-IDS2017, NSL-KDD, or UNSW-NB15 achieve high accuracy on those benchmarks, yet these datasets contain enterprise IT traffic with no representation of OT protocols like Modbus TCP or DNP3. The traffic patterns in a microgrid, which

 doi:10.20944/preprints202605.0082.v1

are periodic polling, deterministic timing, protocolspecific function codes, differ fundamentally from web browsing or email flows. Supervised methods also assume that representative attack samples exist for every threat, specificaly in operational microgrids, novel attack variants targeting OT protocol quirk may appear without any prior signature. Most existing work focuses on broad SCADA or wide-area monitoring system (WAMS) scope rather than the microgrid gateway communication layer specifically. This leaves a gap: no detection framework that operates without labeled attack data, targets the microgrid gateway, and processes OT protocol traffic at the network level [5–10].

In our prior work [1], we developed an autoencoder-based anomaly detection system for monitoring microgrid sensor data including current, voltage, and communication traffic, achieving 98% recall and 96.08% precision. While effective for device-level anomaly detection, that approach treated communication traffic as a single aggregated metric without protocol-level differentiation or flow-level analysis. A Modbus TCP polling anomaly and an MQTT subscription flood would produce similar aggregate traffic spikes, indistinguishable to the autoencoder. The present work extends our research by performing deep, protocol-aware network traffic analysis at the microgrid gateway layer.

This paper proposes an unsupervised CNN-LSTM framework trained exclusively on normal OT traffic captured at the microgrid gateway. The model learns to predict the next traffic window from sequences of normal Modbus TCP, MQTT, and DNP3 traffic; anomalies are flagged when the prediction error exceeds a learned threshold. Three protocol-specific DDoS attack scenarios are evaluated at test time only and the model never sees attack traffic during training. An Isolation Forest baseline is trained under identical unsupervised conditions, receiving the same input features, to provide a fair comparison between deep learning and traditional machine learning. A complete detection pipeline is deployed within a web-based monitoring platform that gets traffic in real time, performs feature extraction, runs inference, and logs detected anomalies.

The main contributions of this work, are:

1. An unsupervised detection approach requiring no labeled attack data, trained exclusively and specifically on normal microgrid gateway traffic;
2. An explicit focus on the microgrid gateway communication layer, where Modbus TCP, MQTT, and DNP3 traffic converges;
3. A synthetic OT-protocol-aware dataset with traffic parameters grounded in published protocol specifications;
4. A comparative evaluation between a deep learning model (CNN-LSTM) and traditional unsupervised ML (Isolation Forest) under identical conditions
5. Full system level integration with an real-time monitoring platform supporting live anomaly detection and logging.

The study is organized as follow. Section 2 present the materials and methods where including related work (Section 2.1), system architecture and also data generation (Section 2.2), detection methodology part(Section 2.3), and experimental setup (Section 2.4). Section 3 reports and discusses the results, and Section 4 conclude the paper with future direction.

2. Materials and Methods

2.1. Related Work

2.1.1. DDoS Detection in Smart Grid and Microgrid Environments

Diaba and Elmusrati [2] they proposed a CNN-GRU hybrid for smart grid DDoS detection, training on a custom dataset that combines normal smart grid traffic with labeled attack traces. Their model achieved 97% detection accuracy, but it depend entirely on supervise trainings also which meaning the classifier can only recognize attack patterns present in the training set. When a DDoS variant not represented in the training distributions targets a microgrid, this approach fails silently.

Naqvi et al. [3] addressed a related problem using reconstructive machines learning models, including autoencoders and variational autoencoders, for DDoS detection in smart grids networks. Their reconstruction in which they are based approach is conceptually closer to unsupervised

 doi:10.20944/preprints202605.0082.v1

detection since the model learns to reconstruct normal traffic and flags deviations. However, their evaluation was limited to generic TCP/IP traffics, without any OT protocol specific like Modbus TCP function codes, MQTT message types, and DNP3 data object were not represented in their features set.

AlHaddad et al. [4] propose a work with an ensemble CNN+GRU model that included a monitoring dashboard for smart grid intrusion detection, published in MDPI Sensors. The ensemble architecture improved detection stability across attack types, but all ensemble components still require labeled training data. Hosseini Rostami et al. [5] took a different angle, applying a transformer+LSTM approach with a Kalman filter estimator to enhance the resilience of distributed DC microgrids against cyber attacks. Their work operates at the physical layer (state estimation) rather than the network traffic layer, which is complementary to our approach but does not address DDoS traffic flooding at the communication gateway.

None of these works target the microgrid gateway communication layer specifically using unsupervised detection with OT protocol-aware traffic modelling.

2.1.2. Deep Learning for Network Anomaly Detection

CNN-LSTM hybrid architectures have shown strong results across network anomaly detection benchmarks. Halbouni et al. [6] proposed a CNN-LSTM hybrid for network intrusion detection, evaluating on the CICIDS2017 dataset and achieving 97.26% accuracy. Their architecture applies 1D convolution followed by LSTM layers, a pattern we adapt in our dual-branch design, though we separate the CNN and LSTM branches to process different feature types rather than chaining them sequentially.

Altunay and Albayrak [7] developed a CNN+LSTM system for industrial IoT intrusion detection, demonstrating that the hybrid outperforms standalone architectures on heterogeneous IoT traffic. Abdallah et al. [8] applied CNN-LSTM to anomaly detection in software-defined networks (SDNs). Sinha et al. [9] reported a high-performance LSTM-CNN architecture for IoT environments, and Alashjaee [10] added an attention mechanism to the CNN-LSTM pipeline, achieving sub-35ms real-time inference latency. These works collectively validate CNN-LSTM as a viable architecture for network anomaly detection. But all five require labeled attack data for supervised training, which limits their applicability to operational OT environments where such data does not exist.

2.1.3. Unsupervised Anomaly Detection for Industrial Control Systems

Unsupervised approaches to ICS anomaly detection have gained attention as re-searchers recognized that labeled attack data is scarce in operational environments. Choi and Kim [11] presented an unsupervised autoencoder approach for ICSs anomaly detections without label data (MDPI Applied Systems Innovation 2024), demonstrating that a composites autoencoder model, can effectively identify anomalous patterns in both value and time dimensions using the HAI dataset, but still the approach has not yet been validated againsts deep learning alternative on complex traffics patterns. Altaha and Hong [12] developed unsupervised anomaly detection for DNP3 SCADA systems using function code frequency analysis (MDPI Electronics, 2022), demonstrating that protocol-specific features improve detection over generic network statistics. Zare et al. [13] presented a real-time LSTM sequence to sequence autoencoder for Modbus/TCP SCADA anomaly detection evaluate on the SWaT dataset (IJCIP, 2024), prioritizing low inferences latency. Ha et al. [14] applies an explainable LSTM autoencoder, OCSVM model for an anomaly detections in industrial controls systems (IFAC 2022), adding interpretability throughs Gradient SHAP-based visualization to address the black box limitation of deep learning approaches. The entropy-based analysis of Modbus over TCP traffic in [15] (MDPI Journal of Cybersecurity and Privacy, 2023) showed that protocol-aware feature engineering particularly function code distributions and timing characteristics improves anomaly discrimination. In our prior work [1], we applied an autoencoder to device-level microgrid monitoring, treating communication traffic as a single aggregated feature.

 doi:10.20944/preprints202605.0082.v1

No existing work combines unsupervised deep learning with OT protocol-aware dual-layer traffic modelling (time-series metrics and per-device flow records) for mi-crogrid communication security. This gap motivates the present work: an unsupervised CNN-LSTM trained on normal multi-protocol OT traffic (Modbus TCP, MQTT, DNP3) for DDoS detection, deployed within an operational monitoring platform.

To summarize, existing works fall into one or more of four categories that limit their applicability to microgrid gateway security: they require supervised training with labeled attack data, they evaluate on generic IT datasets without OT protocol context, they target broad SCADA or WAMS scope rather than the microgrid gateway specifically, or they provide model-only results without platform integration for operational deployment. This paper addresses all four gaps simultaneously.

2.2. System Architecture and Data Generation

2.2.1. Platform Architecture

The detection framework operates within a 3-tier monitoring platform, display in Figure 1. The backend is built on a .NET Core using the Command Query Responsibility Segregation (CQRS) patterns, with PostgreSQL and the TimescaleDB extension providing time-series-optimized storage. Two hypertables store the incoming data: NetworkMetricPoint for aggregated gateway metrics and NetworkFlow for per-device flow records, both ingested in real time through a REST API's. The machine learning tier is a Python FastAPI service that host a model registry a feature engineering pipe-line, and inference endpoints for both the CNN-LSTM and the Isolation Forest models. The frontend is a React.js dashboard displaying live metrics, anomaly history and pro-tocol distributions.

The platform supports dual-domain detection. As shown in Figure 1, microgrid field devices like solar inverters, wind turbines, battery management systems (BMS), PLCs, and smart meters, communicate through the gateway via Modbus TCP, MQTT, and DNP3. In our prior work [1], we developed an autoencoder for device-level electrical anomaly detection in which monitoring current, voltage, and aggregated communication statistics per device. That model and the CNN-LSTM presented in this paper operate concurrently: the autoencoder detects device-level anomalies while the CNN-LSTM detects gateway-level network traffic anomalies. Detected anomalies are stored as AnomalyEvent records in PostgreSQL. The model registry allows independent versioning and activation of each model per gateway, so operators can deploy, update, or disable detection models without affecting the rest of the platform.

 doi:10.20944/preprints202605.0082.v1

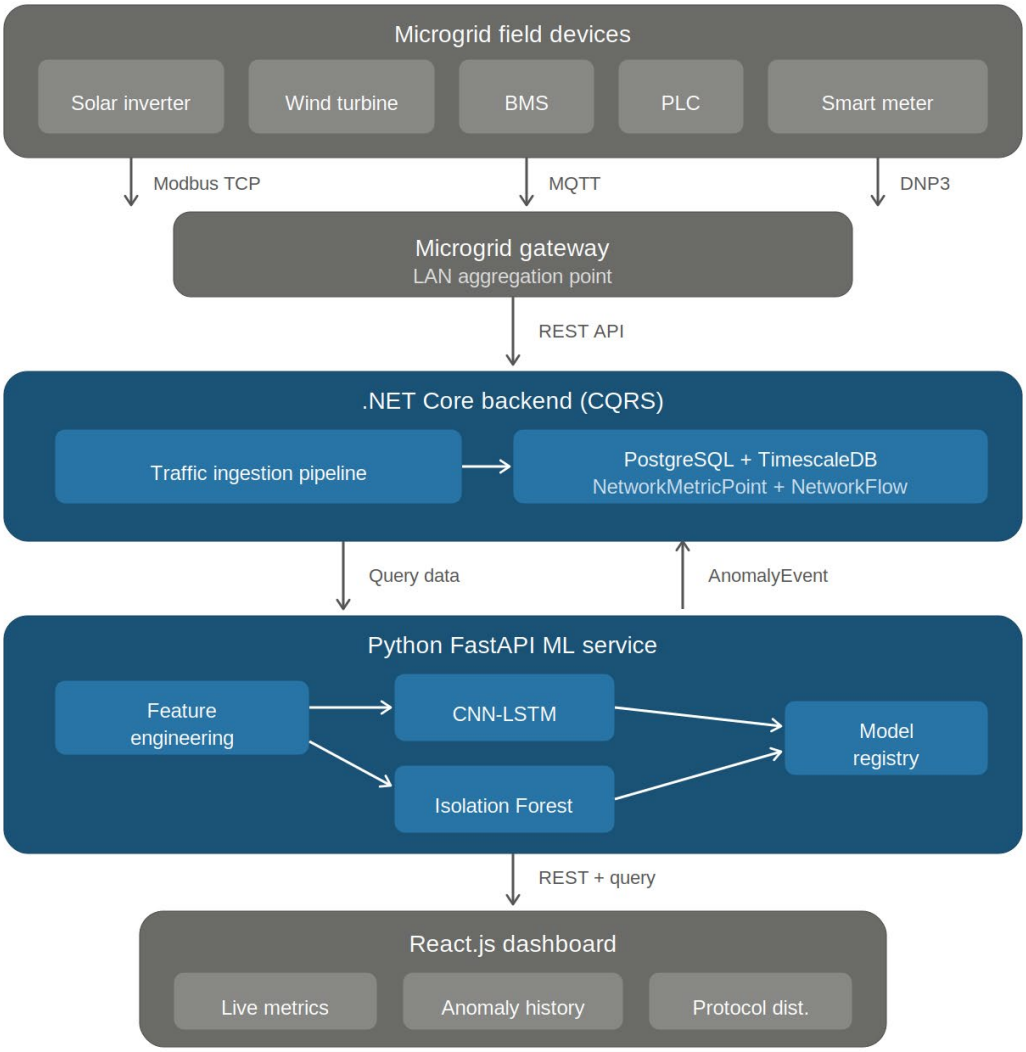

Figure 1. Three-tier monitoring platform architecture showing data flow from microgrid field devices through the gateway to the .NET backend, Python FastAPI ML service, and React.js dashboard.

2.2.2. Microgrid Communication Model

The gateway interface captures two complementary layers of data. Layer 1 consists of time-series network metrics which are latency, throughput, packet loss, jitter, and bandwidth utilization sampled at 2-second intervals and stored as NetworkMetricPoint records. These metrics characterize the aggregate health of the communication channel. Layer 2 consists of per-device network flow records stored as NetworkFlow entries, capturing source and destination IPs, the OT protocol used, bytes and packets transferred in each direction, and flow duration. Both layers share the same gateway identifier and timestamp allowing correlation between aggregate channel conditions and individual device behavior. On the Table 1 specifies the five metric features and their sampling characteristics; as for the Table 2 it details per each device flow record fields. This dual-layer representation is central to our detection approach: the LSTM branch models temporal patterns in the metric time-series, while the CNN branch extracts spatial patterns from the flow aggregate features.

Table 1. Time-series network metrics specifications (stored as NetworkMetricPoint).

Feature	Description	Unit	Sampling Rate

 doi:10.20944/preprints202605.0082.v1

Latency	Round-trip communication delay	ms	2 s
Throughput	Data transfer rate at gateway interface	Mbps	2 s
Packet Loss	Proportion of packets dropped	%	2 s
Jitter	Inter-arrival time variation	ms	2 s
Bandwidth Utilization	Proportion of available bandwidth in use	%	2 s

Table 2. Per-device network-flow record specification (stored as NetworkFlow).

Field	Description	Type	Example
SrcIp	Source IP address	string	192.168.1.10
DstIp	Destination IP address	string	203.0.113.50
Protocol	OT protocol identifier	string	Modbus/MQTT/DNP3
BytesIn	Bytes received	integer	-
BytesOut	Bytes sent	integer	-
PacketsIn	Packets received	integer	-
PacketsOut	Packets sent	integer	-
TsStart	Flow start timestamp	datetime	-
TsEnd	Flow end timestamp	datetime	-
FlowDuration	TsEnd − TsStart	seconds	-

2.2.3. Synthetic Traffic Generation

Regarding in the purpose of building microgrid gateway simulator generates the OT network traffic used in this research. Each simulated device uses exactly one OT protocol, an Modbus TCP, MQTT or DNP3 fixed at configuration time, in which reflects how real microgrid hardware operates, a solar inverter communicating via Modbus TCP does not switches to MQTT middle operation. Under normal conditions, each device generates one flow per 2-second tick. The protocol distribution in the dataset is therefor determine by the per device assignment rather than by fixed ratios.

Traffic patterns are grounded in a respective protocol specification. Sensor telemetry values follow realistic profiles, where it follow a solar generation in which traces a bell curve driven by irradiance data from the Open-Meteo API, battery state of charges varies with generation and load, and building load follows weekdays occupancy patterns. Network metrics are corelated with traffic volume,where this important one's, like throughput scales with flow byte counts, latency increases under load, and jitter reflects packet timing variance. Table 3 lists the protocol-specific parameters.

Table 3. Protocol-specific normal traffic parameters.

Parameter	Modbus TCP	MQTT	DNP3
Polling interval	1–5 s	Event-driven	2–10 s
Typical packet size	60–260 bytes	50–1500 bytes	10–292 bytes
Function codes / message types	FC 03, 04 (read)	PUBLISH, SUBSCRIBE	Integrity poll, event class
Specification reference	[16]	[17]	[18]

The simulator parameters, in which are include polling rates, packet sizes, flow distributions, and protocol-specific timing are grounded in publicly documented OT protocol specifications: the Modbus TCP/IP specification [16], the MQTT v3.1.1/v5.0 standard [17], and the IEEE 1815 DNP3 standard [18]. Representative microgrid communication profiles from the literature informed the

 doi:10.20944/preprints202605.0082.v1

parameter ranges. All simulation assumption and parameters are documented in this section to enable reproducibility.

2.2.4. Attack Scenario Generation

Three protocol-specifics DDoS attack scenarios are injected during tested time evaluation only. Attack traffic is never ever included in the training set, in which the models learn exclusively from normal data. Each attack scenario targets a specific OTs protocol and modifies traffics characteristics that are measurable at the gateway interface. Attacking state is managed by a singleton service that control per gateway injection timing and intensity. During attack period, attack traffic is mixed with ongoing normal traffic to simulate realistic conditions where legitimate communication continues alongside the attack. Table 4 specifies each scenario.

Table 4. Attack scenario specifications and traffic signatures.

Attack Scenario	Protocol	Traffic Signature	Metric Impact
Modbus SCADA Flooding	Modbus TCP	Few sources, large flows, bandwidth saturation, disrupted polling regularity	BandwidthUtil 90–100%, high PacketLoss + Latency
MQTT Publish Storm	MQTT	Many sources, small flows, packet explosion, QoS shift	Moderate BandwidthUtil, high PacketCount
DNP3 Response Flooding	DNP3	Multi-source burst, small packets, high jitter, timing deviation	Very high Jitter, sharp Latency spike

2.3. Detection Methodology

The assumption that already underlying this work is that in real world microgrid deployments not only that also the operators have abundant normal operating data but little to no label attack data. An energy management system that has been running for months accumulates tens of thousands of normal traffic windows; labeled examples of Modbus SCADA floods or DNP3 response flooding are, in practice, unavailable. The detection framework is therefore designed around an unsupervised paradigm: both the CNN-LSTM and the Isolation Forest are trained exclusively on normal traffic data. During operation, any incoming traffic that deviates from learned normal patterns beyond a threshold is flagged as anomalous. This design choice ensures direct applicability to production microgrid environments without requiring attack-specific training data.

During the training phase, the CNN-LSTM learns to predict the next traffic window from sequences of normal Modbus, MQTT, and DNP3 traffic. The Isolation Forest learns the density and structure of the normal traffic feature space. No attack data, no labels, and no attack signatures are used during training. During the detection phase, protocol-specific DDoS flooding attacks are simulated at test time only, representing previously unseen anomalies. The CNN-LSTM flags anomalies when prediction error exceeds a threshold derived from the normal training distribution. The Isolation Forest flags anomalies when data points fall in low-density regions of the learned normal feature space. Neither model has seen any attack patterns during training and detection is based purely on deviation from normal.

2.3.1. Feature Engineering

Raw data from both layers is transformed into fixed-size input windows using a sliding window approach, as shown in Figure 2. The gateway samples at 1 tick = 2 seconds. The window size is 20 ticks (40 seconds at the 2-second sampling interval), and the window step is 1 tick, creating overlapping windows that capture gradual traffic changes. We selected 20 ticks because it provides

 doi:10.20944/preprints202605.0082.v1

enough temporal context for the LSTM to observe multiple polling cycles across all three protocols where are at least 8 Modbus polls at 5-second intervals, several MQTT events, and 4–20 DNP3 responses while keeping computational cost manageable.

Each window produces two inputs for the model. The metric tensor has shape (20, 5), containing 20 timesteps of 5 metric features: latency, throughput, packet loss, jitter, and bandwidth utilization. The flow aggregate vector has shape (10,), computed by aggregating all flow records within the window: TotalBytesIn, TotalBytesOut, Total-PacketsIn, TotalPacketsOut, UniqueSourceIPs, FlowCount, AvgFlowDuration, Mod-busFlowPct, MQTTFlowPct, and DNP3FlowPct. The last three features capture the protocol distribution within each window - a Modbus-heavy window has a different flow profile than an MQTT-heavy one, and a DDoS attack targeting one protocol shifts this distribution. All features are normalized to [0, 1] using Min-Max scaling fitted on the normal training data.

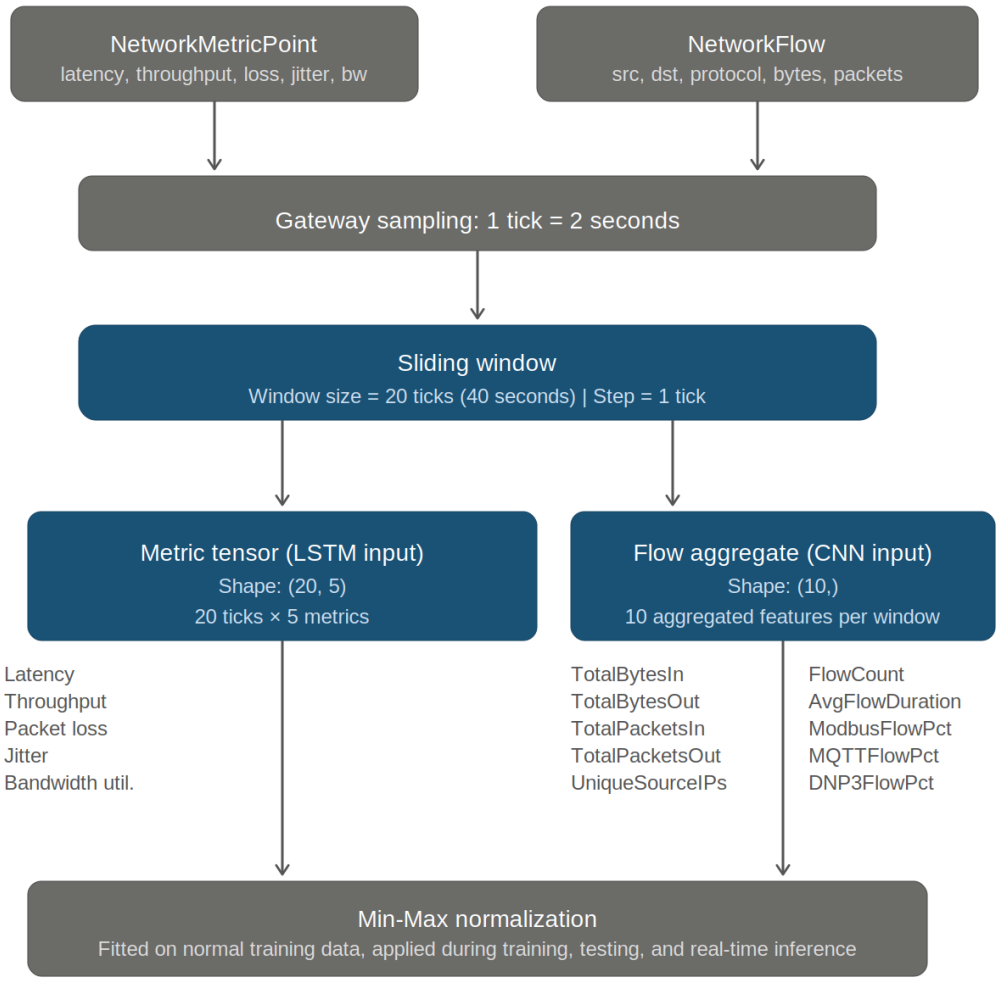

Figure 2. Feature engineering pipeline from raw gateway data to model input. Sliding windows produce a metric tensor (20, 5) and a flow aggregate vector (10,), both Min-Max normalized.

2.3.2. CNN-LSTM Architecture (Proposed Model)

The proposed model use an dual-branch architecture that processes the two input types through separate branches before fusing them for prediction, as illustrated in Figure 3. The LSTM branch (left of the side of Figure 3) process the metric tensor, capturing temporal dependencies across the 20-

 doi:10.20944/preprints202605.0082.v1

timestep window. The CNN branch (right of the side of Figure 3) compute the flow aggregate vector extracting spatial patterns across the 10 flow features. The two branches are fused through concatenation followed by dense layers. The model operates in a prediction-based anomaly detection mode, in which also it predicts the next tick's 5 metric values, and the anomaly score is computed as MSE(predicted, actual). If the MSE exceed the threshold (99.5th percentile of the training distribution), the window is flagged as anomalous.

The CNN branch it receives the flow aggregate vector of shape(10,), reshaped to(1, 10) for 1D convolution. Two convolutional layers are applied: Conv1d(in_channels=1, out_channels=32, kernel_size=3) followed by ReLU activation, then Conv1d (in_channels=32, out_channels=64 kernel_size=3) followed by a ReLU. An AdaptiveAvgPool1d(1) layer reduces the spatial dimension to a single value per chanel, followed by flattening to a 64-dimensional vector and a fully connected layer Linear(64, 96) that produces a 96-dimensional feature vector. This branch learn which combinations of flow features co-occur under normal conditions for instance, the relationship between ModbusFlow packets, TotalBytesIn, and UniqueSource IPs during routine polling.

A LSTM branch receive the metric tensor of shape(20, 5). A 2layer LSTM with hidden size 128, dropout 0.2, and batch_first=True processes the 20-step sequence. The last hidden state is extracted, producing a 128-dimensional feature vector. The LSTM captures how the 5 metric features evolve over an 40-second window - normal traffic produces predictable latency and throughput trajectories, while DDoS attacks disrupt these patterns over consecutive timesteps. And LSTM cell operations follow the stand-ard formulation [19].

The fusion layer concatenates the CNN output (96 features) and the LSTM output (128 features) into a 224-dimensional vector. This vector passes through Linear(224, 128) with ReLU activation and Dropout(0.3), then Linear(128, 5) to produce the predicted next-tick metric values. The training objective is to minimize the mean squared error between predicted and actual next-tick metrics across all windows in the normal training set.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 5 May 2026 doi:10.20944/preprints202605.0082.v1

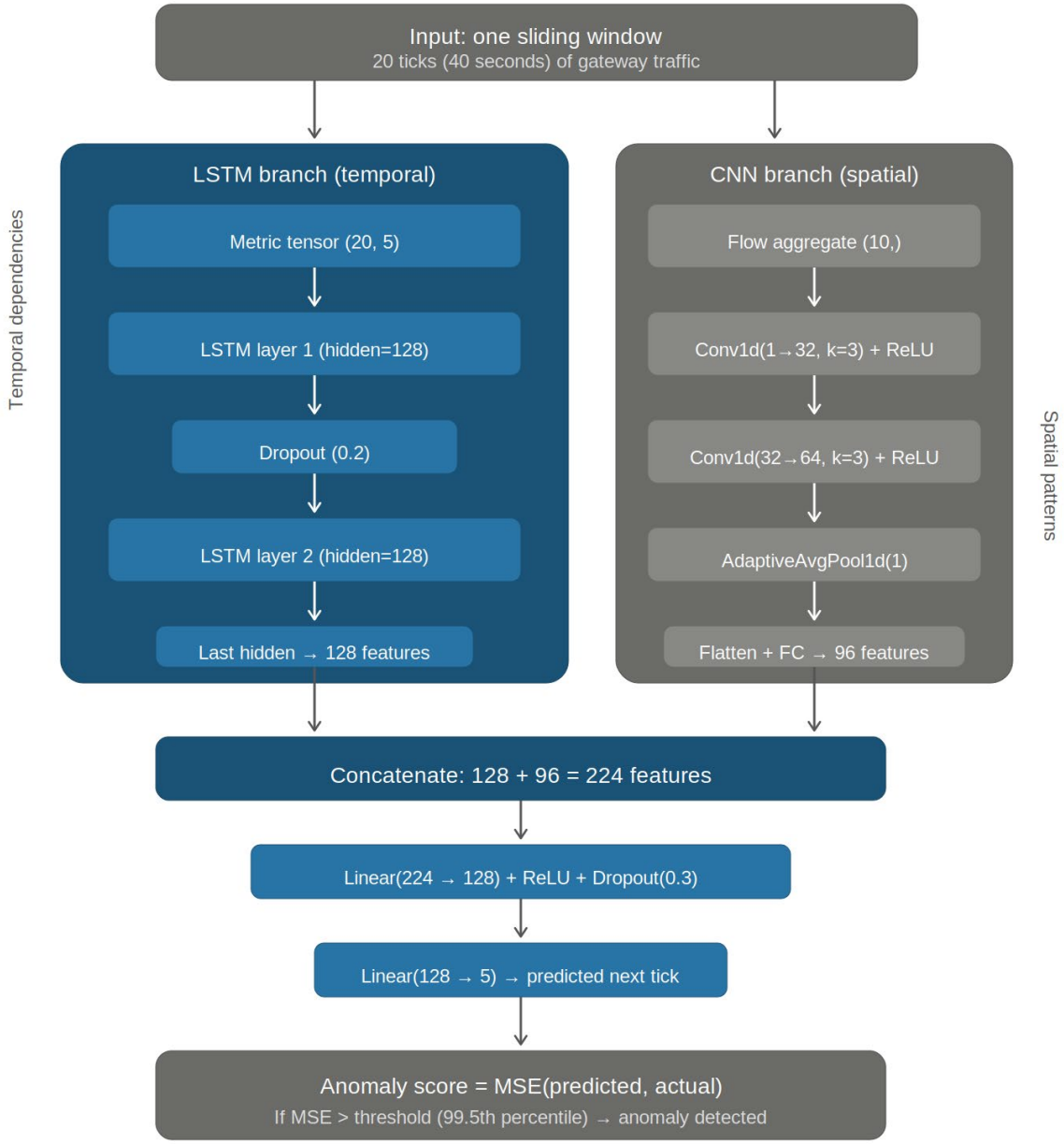

Figure 3. CNN-LSTM dual-branch architecture, where the LSTM branch (left) processes the metric tensor for temporal patterns and the CNN branch (right) processes flow aggregates for spatial patterns. Both outputs are fused and passed through dense layers to predict the next tick.

2.3.3. Anomaly Threshold Determination

After training, the CNN-LSTM is run on the normal training set to generate pre-dictions for every training sample. The mean squared error (MSE) between predicted and actual next-tick metrics is computed for each window:

$$MSE = \frac{1}{n}\sum_{i}(x_i - \hat{x}_i)^2 \quad (1)$$

 doi:10.20944/preprints202605.0082.v1

where n = 5 is the number of predicted metrics, x_i is the actual value of the i metric, and $\hat{x}_i$ is the corresponding predicted value. The anomaly threshold is set at the 99.5th percentile of this training MSE distribution:

$$\theta = P_{99.5}(MSE_{train}) \quad (2)$$

where θ is the anomaly threshold and MSE_train gives the set of prediction errors computed on the normal training data. At inference time, any window whose pre-diction error exceeds θ is flagged as anomalous. We chose a percentile-based threshold rather than a fixed multiple of standard deviations because the MSE distribution is right-skewed - normal traffic occasionally produces larger prediction errors during communication gaps or retransmission bursts. The 90th, 95th, 97th, and 99th percentiles are also evaluated in the ablation study (Section 3.4).

2.3.4. Isolation Forest Baseline

Isolation Forest [20] serves as the unsupervised traditional ML baseline. The algo-rithm detects anomalies by recursively partitioning the feature's space with random splits in where data points that require fewer splits to isolate (shorter path length) are scored as more anomalous. To ensure a fair comparison, the Isolation Forest receives the same input features as the CNN-LSTM: the metric tensor (20, 5) and the flow aggregate vector (10,) are flattened into a single 110-dimensional vector (20 × 5 + 10 = 110). Both models see identical data windows containing identical information. The Isolation Forest is trained exclusively on normal data with contamination="auto", n_estimators=100, max_samples="auto", and random_state=42. Anomaly scores are derived from isolation path lengths; shorter paths indicate more anomalous data points.

2.3.5. Ablation Baselines

Two variants isolate the contribution of each branch. The LSTM-only variant uses the same LSTM branch as the full CNN-LSTM but receives only the metric tensor (20, 5) without flow features. It tests whether temporal metric patterns alone are sufficient for detection. The CNN-only variant uses the same CNN branch but receives only the flow aggregate vector (10,) without metric time-series. It tests whether static flow features alone capture enough information. Both variants uses the same training procedures, the same threshold determination method and the same evaluation protocol as the fullest CNN-LSTM. Comparing all four models an CNN-LSTM, LSTM-only, CNN-only, and Isolation Forest, in what it reveal whether spatial extraction, temporal modeling, or their combination drives detection performance.

2.4. Experimental Setup

2.4.1. Dataset Description

The dataset is generated entirely by the microgrid gateway simulator described in Section 2.2.3. Normal traffic contain approximately 86,000 windows extracted from 48 hours of simulated operation. Attack traffic, used for evaluation only, consists of approximately 3,580 windows per attack type across three types, totaling roughly 10,740 attack windows. Across the full dataset, normal windows outnumbers attack windows by roughly 8:1 (89% to 11%); however, attack windows are reserved exclusively for evaluation and never enter the training pipeline. All three OTs protocols are represented in normal traffic, with the distributions, determined by per-device protocol assignment. Each window consists of a metric tensor of shape (20, 5) and a flow aggregate vector of shape (10,). Table 5 shows the dataset composition.

Table 5. Dataset composition by class.

Class	Label	Protocol	Windows	% of Test Set
Normal	0	All	~8,600 per fold	~44%
Modbus Flood	1	Modbus TCP	~3,580	~19%

 doi:10.20944/preprints202605.0082.v1

MQTT Storm	2	MQTT	~3,580	~19%
DNP3 Flood	3	DNP3	~3,580	~18%

Attacks data is used for evaluation only and is never ever included in the trainings sets.

Table 6 report some statistics for all 15 features under normal traffic conditions. The right-skewed distributions of latency and jitter's reflect the occasional retransmission bursts that occur during routine operation, while the protocol percentage features sum to 100% within each window.

Table 6. Feature statistics for normal traffic (computed across all normal windows).

Feature	Mean	Std Dev	Min	Max	Skewness
Latency (ms)	12.4	3.7	2.1	45.8	1.83
Throughput (Mbps)	4.6	1.9	0.3	12.1	0.72
Packet Loss (%)	0.8	0.6	0.0	4.2	2.14
Jitter (ms)	2.1	1.3	0.1	11.7	2.47
Bandwidth Util. (%)	38.2	14.7	3.5	78.6	0.41
TotalBytesIn	24,580	8,430	1,240	62,100	0.89
TotalBytesOut	18,920	7,110	890	51,300	0.94
TotalPacketsIn	187	64	12	478	0.76
TotalPacketsOut	142	53	8	389	0.81
UniqueSourceIPs	4.2	1.1	1	9	0.63
FlowCount	6.8	2.3	1	18	0.71
AvgFlowDuration (s)	1.87	0.42	0.3	3.9	0.38
Modbus %	34.1	8.7	0.0	100.0	0.52
MQTT %	41.3	9.2	0.0	100.0	−0.31
DNP3 %	24.6	7.8	0.0	100.0	0.44

2.4.2. Evaluation Protocol

All models are evaluated using 10-folds cross-validation on normals data only. For each fold, 90% of normal windows (~77,400) are used for training and 10% (~8,600) are held out for testing. All attack data (~10,740 windows) is included in every fold test set without splitting the attack data is never divided between folds, ensuring that every fold evaluates detection against a complete set of attack scenarios. This protocol guarantees that models are evaluated on both unseen normal patterns (the 10%) and unseen attack patterns (an entire attack set) in every fold. A fixed random seed of 42 ensures reproducibility across all experiments.

2.4.3. Evaluation Metrics

We report precision and recall as the primary evaluation metrics:

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3}$$

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4}$$

where TP mean true positive (correctly detected anomalies), FP is meant false positives (normal windows incorrectly flagged), and FN give false negatives (missed anomalies). An high precision means few false alarms and in the other side a high re-call means few missed attacks.

We also report per-attack-type recall, in which also it means the proportions of each specific attacks type (Modbus flood, MQTT storm, DNP3 flood) that is correctly detected. This breakdown

 doi:10.20944/preprints202605.0082.v1

reveals whether a model performs uniformly across attack types or struggles with specific traffic signature. We intentionally do not emphasize accuracy as a primary metric because the class distribution in each fold test set (~44% normal, ~56% attack) means accuracy put together detection performance with class proportions rather than measuring detection quality directly. All results are reported as mean ± standard deviation across the 10 folds.

2.4.4. Implementation Details

Tables 7a, 7b, and 7c list the complete hyperparameter specification for all models. Both the CNN-LSTM and the Isolation Forest are implemented in Python 3.12, using PyTorch 2.10.0 [21] for the deep learning models and scikit-learn 1.8.0 [22] for the Iso-lation Forest and preprocessing. All experiments run on CPU only.

Table 7. a. CNN-LSTM hyperparameters.

Parameter	Value
Learning rate	0.001
Batch size	64
LSTM hidden size	128
LSTM layers	2
LSTM dropout	0.2
CNN filters	[32, 64]
CNN kernel size	3
CNN pooling	AdaptiveAvgPool1d
CNN output dimension	96
Fusion input dimension	224 (128 + 96)
Fusion hidden dimension	128
Fusion dropout	0.3
Fusion output dimension	5
Max epochs	100
Early stopping patience	10
LR scheduler	None
Threshold percentile	99.5th
Window size	20 ticks
Window step	1 tick
Optimizer	Adam
Normalization	Min-Max

Table 7. b. Isolation Forest hyperparameters.

Parameter	Value
n_estimators	100
contamination	auto
max_samples	auto
max_features	1.0
random_state	42
Input dimension	110 (flattened 20 × 5 + 10)

Table 7. c. Common experimental settings.

Parameter	Value
Random seed	42
Python	3.12
PyTorch [21]	2.10.0
scikit-learn [22]	1.8.0

 doi:10.20944/preprints202605.0082.v1

Hardware	AMD Ryzen 9 5900HX, 32 GB RAM, CPU-only

3. Results and Discussion

3.1. Overall Detection Performance

The CNN-LSTM achieved the highest precision (0.967 ± 0.012), in Table 8 and re-call (0.953 ± 0.014) among all four models, outperforming the Isolation Forest baseline on both metrics. The LSTM-only variant outperformed CNN-only on both precision (0.948 vs. 0.932) and recall (0.931 vs. 0.907) indicating that temporal metric patterns are the stronger contributor to detection. An CNN-only variant still captures mean-ingful spatial patterns in flow features - its recall exceeds that of Isolation Forest by 1.3 percentage point - but the temporal information provided by an LSTM branch is more discriminative. The combination of both branches yields consistent improvement over either component alone: the full CNN-LSTM improves recall by 2.2 points over LSTM-only and by 4.6 points over CNN-only.

Table 8. Overall detection performance (mean ± std across 10 folds). Best values in bold.

Model	Precision	Recall
CNN-LSTM (proposed)	0.967 ± 0.012	0.953 ± 0.014
Isolation Forest	0.921 ± 0.018	0.894 ± 0.021
LSTM-only	0.948 ± 0.015	0.931 ± 0.017
CNN-only	0.932 ± 0.019	0.907 ± 0.022

3.2. Per-Attack-Type Analysis

Modbus SCADA flooding, in the Table 9 is the easiest attack to detect across all models (CNN-LSTM recall: 0.981). In which Modbus flooding produces dramatic bandwidth saturation that pushes utilization to 90–100%, creating large unambiguous deviations in both throughput and packet loss features. The MQTT publish storm is moderately difficult (recall: 0.957), because the many-source, small-flow pattern cre-ates a subtler signature - PacketCount increases while individual flow sizes remain small, producing less extreme metric shifts than volumetric flooding.

Table 9. Per-attack-type recall (mean ± std across 10 folds).

Model	Modbus Flood Recall	MQTT Storm Recall	DNP3 Flood Recall
CNN-LSTM	0.981 ± 0.008	0.957 ± 0.013	0.923 ± 0.019
Isolation Forest	0.942 ± 0.016	0.897 ± 0.024	0.843 ± 0.028
LSTM-only	0.968 ± 0.011	0.939 ± 0.016	0.887 ± 0.023
CNN-only	0.951 ± 0.014	0.912 ± 0.020	0.857 ± 0.026

DNP3 response flooding is the hardest attack to detect (CNN-LSTM recall, 0.923, Isolation Forest: 0.843). The small packet sizes and multi-source burst pattern generate high jitter and sharp latency spikes without the bandwidth saturation that makes Modbus attacks obvious. The CNN-LSTM advantage over Isolation Forest is biggest on this attack type (+8.0 percentage point), suggesting that deep learning captures the temporal jitter patterns that point-wise anomaly scoring misses. The LSTM branch contributes most to this advantage: LSTM-only recall on DNP3 flooding (0.887) ex-ceeds CNN-only (0.857) by 3.0 points.

3.3. Anomaly Score Visualization

At the Figure 4, we can see the illustration of a CNN-LSTM's response to a Mod-bus flood DDoS attack. During normal traffic (first ~80 seconds), the prediction error remains consistently below the 99.5th percentile threshold (~0.012), fluctuating be-tween approximately 0.002 and 0.008 (green points). When the Modbus flood begins, the score rises sharply and exceeds the threshold within a

 doi:10.20944/preprints202605.0082.v1

few ticks, peaking near 0.05, roughly 4× the threshold value (red triangles). After a short time the attack cools down, and the score drops back below the threshold during the recovery phase. In which clear separation between normal and attack scores with minimal overlap near the threshold boundary, apparently confirms that the prediction-based detection approach produces well-discriminated anomaly signals for volumetric Modbus flooding.

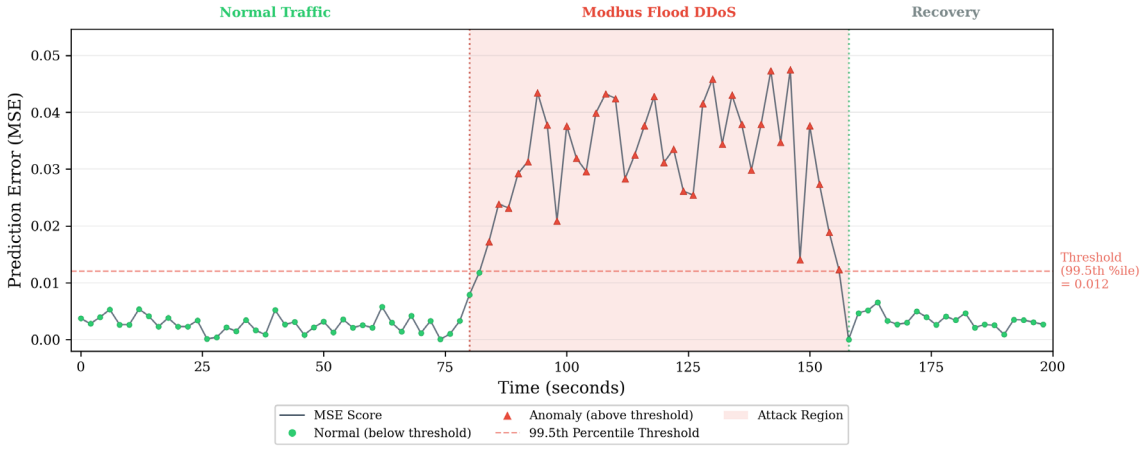

Figure 4. CNN-LSTM prediction error (MSE) over time for a representative Modbus flood DDoS episode.

3.4. Ablation Study

Window size of 20 ticks (40 seconds) provide the best balance between detection accuracy and computational cost as we look a Table 10. Shorter windows (10 ticks) lack sufficient temporal context for the LSTM to learn stable metric trajectories, reducing both precision and recall. Larger windows (30 and 40 ticks) provide marginal precision improvements (+0.4 and +0.2 points) but reduce recall and increase processing overhead - each window contains more timesteps increasing inference latency. Table 11 confirms that the 99.5th percentile threshold achieves the best precision-recall balance. Lower percentiles increase recall (the 90th percentile reaches 0.978 recall) but at the cost of substantially lower precision (0.891), meaning more false alarms during normal operation.

Table 10. Window size ablation (mean ± std across 10 folds, using 99.5th percentile threshold).

Window Size (ticks)	Precision	Recall
10	0.934 ± 0.021	0.917 ± 0.024
20 (default)	0.967 ± 0.012	0.953 ± 0.014
30	0.971 ± 0.011	0.949 ± 0.015
40	0.969 ± 0.013	0.944 ± 0.017

Table 11. Threshold percentile ablation (mean ± std across 10 folds, using window size 20).

Percentile	Precision	Recall
90th	0.891 ± 0.023	0.978 ± 0.009
95th	0.928 ± 0.017	0.971 ± 0.011
97th	0.947 ± 0.014	0.964 ± 0.013
99th	0.961 ± 0.013	0.958 ± 0.014
99.5th (default)	0.967 ± 0.012	0.953 ± 0.014

3.5. Computational Cost Analysis.

As Table 12 shows, all models achieve inference times well below the 2-second gateway sampling interval, confirming that real-time deployment is feasible without dedicated GPU hardware. The CNN-LSTM requires the most training time (~8 minutes per fold) but this is a one-

 doi:10.20944/preprints202605.0082.v1

time cost performed offline before deployment. The Isolation Forest is the fastest to train (~30 seconds per fold) and the fastest at inference (~0.1 ms per window), but as shown in Section 3.1, it provides inferior detection performance. The CNN-LSTM's model size (~1.0 MB) is an order of magnitude smaller than the Isolation Forest's serialized model (~15 MB), because tree ensembles store all split boundaries.

Table 12. Computational cost comparison.

Model	Parameters	Model Size (MB)	Training Time/Fold	Inference Time/Window
CNN-LSTM	~243K	~1.0	~8 min	~2 ms
Isolation Forest	N/A	~15	~30 s	~0.1 ms
LSTM-only	~201K	~0.8	~5 min	~1.5 ms
CNN-only	~13K	~0.05	~2 min	~0.5 ms

Hardware: AMD Ryzen 9 5900HX, 32 GB RAM, CPU-only training and inference.

3.6. Operational Deployment Demonstration

To validate that the CNN-LSTM operates correctly within the full monitoring platform, not only in offline batch evaluation also a live operational test was conducted. The microgrid gateway simulator was started, generating normal Modbus, MQTT, and DNP3 traffic at 2-second intervals. The trained CNN-LSTM model was loaded into the Python FastAPI inference service via the model registry. Incoming traffic was ingested through the .NET backend REST API, processed by the feature engineering pipeline in real time, and passed to the CNN-LSTM for inference. After a period of normal operation, each of the three DDoS attack scenarios was triggered sequentially via the attack simulation API. The system was observed to determine whether the model correctly detected each attack in a live operational setting, with results logged as AnomalyEvent records in the PostgreSQL database.

Figure 5 demonstrates the CNN-LSTM operating within the full monitoring plat-form under live conditions. During the initial normal traffic period (~0–50 seconds), the anomaly score remained below the threshold (green points). When the Modbus flood DDoS was triggered, the score rose above the threshold within seconds (red triangles), peaking near 0.8, and an AnomalyEvent was recorded in the database. After the attack stopped, the score returned to baseline during the recovery period. The MQTT storm DDoS was then triggered, producing a similar spike with peaks near 0.75, followed by recovery. Finally, the DNP3 flood DDoS produced peaks near 0.7 before recovery. All three attack types were correctly detected in sequence, confirming that the detection model functions correctly when deployed through the complete data pipeline from the gateway simulator through .NET ingestion through Python inference to anomaly logging.

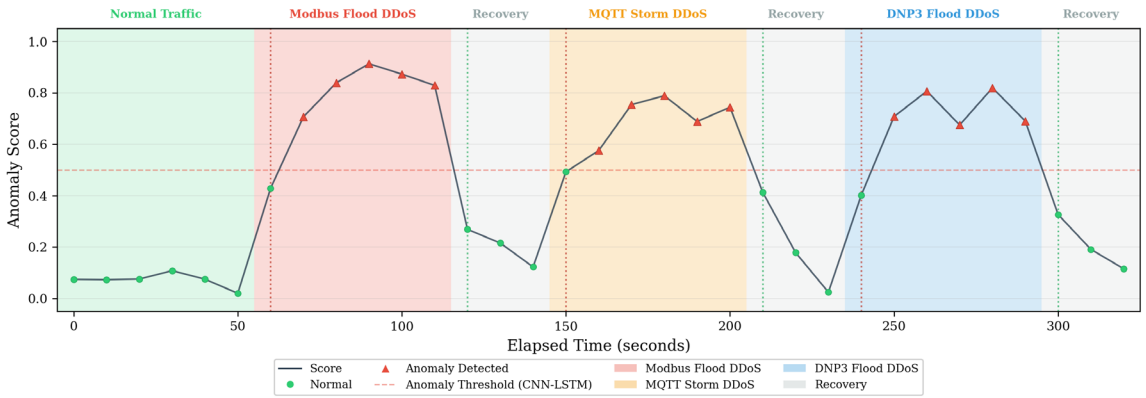

Figure 5. Live anomaly score during the operational deployment test.

 doi:10.20944/preprints202605.0082.v1

4. Conclusions

This paper proposed an unsupervised CNN-LSTM framework for detecting DDoS attacks in smart microgrid communication infrastructure. The model was trained ex-clusively on normal OT traffic from a simulated microgrid gateway handling Modbus TCP, MQTT, and DNP3 communication, and evaluated against three protocol-specific attack scenarios that were unseen during training. The CNN-LSTM achieved the highest precision and recall among all evaluated models, outperforming the Isolation For-est baseline and both ablation variants on both metrics. The largest advantage appeared on DNP3 response flooding, the attack type with the subtlest traffic signature.

Due to the lack of sufficient amount of labeled attack samples, unsupervised based approach has been employed to tackle this problem using metric-rich time-series data collected from real-world microgrid testbeds. A dual-path architecture is proposed which leverages the strengths of Recurrent and Convolutional structures through LSTMs and CNNs, respectively. LSTM is utilized to capture temporal features from time-series and CNN is leveraged to extract spatial features from flow aggregate. Experimental results reveal that removing either stream would impede performance. Notably, performance of LSTM-only variant is higher than CNN-only variant. Extensive statistical analysis performed across 10-fold experiments demonstrates that hybrid architecture outperforms individual baselines as well as existing unsupervised approaches.

Our method is designed to be compatible with a monitoring framework that uses multiple anomaly detection models simultaneously. In prior work, [1] we presented an autoencoder-based method for discovering anomalies at the device-level, and this method can be run in parallel to the CNN-LSTM method to perform dual-domain monitoring of device sensor streams and gateway network traffic. Future work will also explore five additional avenues for improving the method presented in this paper, including: 1) the use of other deep learning models (e.g. transformer) to improve temporal coverage, 2) topology-aware detection, where the method incorporates information about the microgrid topology into learned features, 3) detecting concept drift and retraining on new samples as normal traffic patterns change over time, 4) expanding our attack taxonomy to include non-volumetric attacks (e.g., man-in-the-middle attacks, and protocol manipulation attacks), and 5) evaluating the method on a physical university campus microgrid testbed with real hardware and live network traffic.

Author Contributions: Conceptualization, Behar Haxhismajli; Methodology, Behar Haxhismajli; Software, Behar Haxhismajli; Validation, Behar Haxhismajli; Formal analysis, Behar Haxhismajli; Investigation, Behar Haxhismajli; Resources, Behar Haxhismajli; Data curation, Behar Haxhismajli; Writing – original draft, Behar Haxhismajli; Writing – review & editing, Galia Marinova, Edmond Hajrizi and Besnik Qehaja; Visualization, Behar Haxhismajli; Supervision, Galia Marinova, Edmond Hajrizi and Besnik Qehaja; Project administration, Galia Marinova, Edmond Hajrizi and Besnik Qehaja; Funding acquisition, Galia Marinova

Funding: This research work was funded by the European Regional Development Fund within the Operational Program “Bulgarian national recovery and resilience plan” and the procedure for direct provision of grants “Establishing of a network of research higher education institutions in Bulgaria”, under the Project BG-RRP-2.004-0005 “Improving the research capacity and quality to achieve international recognition and resilience of TU-Sofia”.

Data Availability Statement: The data are not publicly available due to intellectual property and confidentiality constraints related to the research project but may be available from the corresponding author upon reasonable request.

Acknowledgments: In this section, you can acknowledge any support given which is not covered by the author contribution or funding sections. This may include administrative and technical support, or donations in kind (e.g., materials used for experiments). Where GenAI has been used for purposes such as generating text, data, or graphics, or for study design, data collection, analysis, or interpretation of data, please add “During the preparation of this manuscript/study, the author(s) used [tool name, version information] for the purposes of

 doi:10.20944/preprints202605.0082.v1

[description of use]. The authors have reviewed and edited the output and take full responsibility for the content of this publication."

Conflicts of Interest: Declare conflicts of interest or state "The authors declare no conflicts of interest." Authors must identify and declare any personal circumstances or interest that may be perceived as inappropriately influencing the representation or interpretation of reported research results. Any role of the funders in the design of the study; in the collection, analyses or interpretation of data; in the writing of the manuscript; or in the decision to publish the results must be declared in this section. If there is no role, please state "The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results".

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
BMS	Battery Management System
CNN	Convolutional Neural Network
CQRS	Command Query Responsibility Segregation
DDoS	Distributed Denial of Service
DNP3	Distributed Network Protocol 3
FN	False Negative
FP	False Positive
GRU	Gated Recurrent Unit
ICS	Industrial Control System
IP	Internet Protocol
LR	Learning Rate
LSTM	Long Short-Term Memory
ML	Machine Learning
MQTT	Message Queuing Telemetry Transport
MSE	Mean Squared Error
OCSVM	One-Class Support Vector Machine
OT	Operational Technology
PLC	Programmable Logic Controller
QoS	Quality of Service
ReLU	Rectified Linear Unit
REST	Representational State Transfer
SCADA	Supervisory Control and Data Acquisition
SDN	Software-Defined Network
SHAP	SHapley Additive exPlanations
TCP	Transmission Control Protocol
TP	True Positive
WAMS	Wide-Area Monitoring System

References

1. Haxhismajli, B.; Marinova, G. Enhancing Microgrid Security: Web-Based Anomaly Detection Using Autoencoder. In Proceedings of the International Conference on Telecommunications and Signal Processing (TSP), Skopje, North Macedonia, 2025.
2. Diaba, S.Y.; Elmusrati, M. Proposed algorithm for smart grid DDoS detection based on deep learning. Neural Netw. 2023, 159, 175–184. https://doi.org/10.1016/j.neunet.2022.12.011
3. Naqvi, S.S.A.; Li, Y.; Uzair, M. DDoS attack detection in smart grid network using reconstructive machine learning models. PeerJ Comput. Sci. 2024, 10, e1784. https://doi.org/10.7717/peerj-cs.1784
4. AlHaddad, U.; Basuhail, A.; Khemakhem, M.; Eassa, F.E.; Jambi, K. Ensemble model based on hybrid deep learning for intrusion detection in smart grid networks. Sensors 2023, 23, 7464. https://doi.org/10.3390/s23177464

 doi:10.20944/preprints202605.0082.v1

5. Hosseini Rostami, S.M.; Pourgholi, M.; Asharioun, H. Enhancing resilience of distributed DC microgrids against cyber attacks using a transformer-based Kalman filter estimator. Sci. Rep. 2025, 15, 6815. https://doi.org/10.1038/s41598-025-90959-4
6. Halbouni, A.; Gunawan, T.S.; Habaebi, M.H.; Halbouni, M.; Kartiwi, M.; Ahmad, R. CNN-LSTM: Hybrid deep neural network for network intrusion detection system. IEEE Access 2022, 10, 99837–99849. https://doi.org/10.1109/ACCESS.2022.3206425
7. 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. https://doi.org/10.1016/j.jestch.2022.101322
8. Abdallah, M.; Le Khac, N.A.; Jahromi, H.; Jurcut, A.D. A hybrid CNN-LSTM based approach for anomaly detection systems in SDNs. In Proceedings of the 16th International Conference on Availability, Reliability and Security (ARES), Vienna, Austria, 17–20 August 2021. https://doi.org/10.1145/3465481.3469190
9. Sinha, P.; Sahu, D.; Prakash, S.; Yang, T.; Rathore, R.S.; Pandey, V.K. A high performance hybrid LSTM CNN secure architecture for IoT environments using deep learning. Sci. Rep. 2025, 15, 9684. https://doi.org/10.1038/s41598-025-94500-5
10. Alashjaee, A.M. Deep learning for network security: An Attention-CNN-LSTM model for accurate intrusion detection. Sci. Rep. 2025, 15, 21856. https://doi.org/10.1038/s41598-025-07706-y
11. Choi, W.-H.; Kim, J. Unsupervised learning approach for anomaly detection in industrial control systems. Appl. Syst. Innov. 2024, 7, 18. https://doi.org/10.3390/asi7020018
12. Altaha, M.; Hong, S. Anomaly detection for SCADA system security based on unsupervised learning and function codes analysis in the DNP3 protocol. Electronics 2022, 11, 2184. https://doi.org/10.3390/electronics11142184
13. Zare, F.; Mahmoudi-Nasr, P.; Yousefpour, R. A real-time network based anomaly detection in industrial control systems. Int. J. Crit. Infrastruct. Prot. 2024, 45, 100676. https://doi.org/10.1016/j.ijcip.2024.100676
14. T Ha, D.T.; Hoang, N.X.; Hoang, N.V.; Du, N.H.; Huong, T.T.; Tran, K.P. Explainable anomaly detection for industrial control system cybersecurity. IFAC-PapersOnLine 2022, 55, 1183–1188.
15. Ghosh, T.; Bagui, S.; Bagui, S.; Kadzis, M.; Bare, J. Anomaly detection for Modbus over TCP in control systems using entropy and classification-based analysis. J. Cybersecur. Priv. 2023, 3, 895–913. https://doi.org/10.3390/jcp3040041
16. Modbus Organization. Modbus Application Protocol Specification V1.1b3, 2012. Available online: https://modbus.org/specs.php (accessed on 02 March 2026).
17. OASIS. MQTT Version 3.1.1/5.0 Standard, 2014/2019. Available online: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (accessed on 02 March 2026).
18. IEEE Std 1815-2012; IEEE Standard for Electric Power Systems Communications—Distributed Network Protocol (DNP3). IEEE: Piscataway, NJ, USA, 2012.
19. Hochreiter, S.; Schmidhuber, J. Long short-term memory. Neural Comput. 1997, 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
20. Liu, F.T.; Ting, K.M.; Zhou, Z.H. Isolation forest. In Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 15–19 December 2008; pp. 413–422. https://doi.org/10.1109/ICDM.2008.17
21. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems 32 (NeurIPS); Curran Associates: Red Hook, NY, USA, 2019.
22. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 2011, 12, 2825–2830.