**FACULTY OF ENGINEERING, COMPUTING AND THE ENVIRONMENT**

**School of Computer Science and Mathematics**

**Kingston University London**

**MSc Data Science**

Muhammad Sameer

K2453955

Behavior-Based Intrusion Detection in Electric Vehicle Charging Systems: Comparing Machine and Deep Learning Models with Feature Selection on the CICEVSE2024

Date:18/9/2025

Supervisor: Dr Xing Liang

KU London Logo

**WARRANTY STATEMENT**

This is a student project. Therefore, neither the student nor Kingston University makes any warranty, express or implied, as to the accuracy of the data or conclusion of the work performed on the project and will not be held responsible for any consequences arising out of any inaccuracies or omissions therein.

Contents

[I. INTRODUCTION 4](#_Toc209309685)

[II. Related Work 2](#_Toc209309686)

[III. Dataset 2](#_Toc209309687)

[A. Kernel Event Features Space 3](#_Toc209309688)

[B. Data Preprocessing 3](#_Toc209309689)

[C. Dataset Statistics 4](#_Toc209309690)

[IV. Methodology 4](#_Toc209309691)

[A. Baseline Machine Learning Models 4](#_Toc209309692)

[B. Temporal Attack Patterns 4](#_Toc209309693)

[C. Sliding Window Transformation 5](#_Toc209309694)

[D. Model Design 5](#_Toc209309695)

[E. Training Strategy and Evaluation 7](#_Toc209309696)

[F. Evaluation of Metrics. 7](#_Toc209309697)

[V. Results and Analysis 7](#_Toc209309698)

[A. Overall Model Comparison 7](#_Toc209309699)

[B Deep Learning Model Performance 8](#_Toc209309700)

[C Per-Class Performance 8](#_Toc209309701)

[D Key Findings: 8](#_Toc209309702)

[E Error Analysis 8](#_Toc209309703)

[F Comparative Discussion 8](#_Toc209309704)

[VI. Discussion 9](#_Toc209309705)

[A. Impact of Preprocessing 9](#_Toc209309706)

[B. Limitations 9](#_Toc209309707)

[C. Potential Improvements 9](#_Toc209309708)

[VII. Conclusion 9](#_Toc209309709)

[Appendix 10](#_Toc209309710)

[References 10](#_Toc209309711)

[Abbreviations and Acronyms 10](#_Toc209309712)

[Table 1MACHINE LEARNING MODELS EVALUATION 4](#_Toc209307493)

[Table 2 DEEP LEARNING MODELS EVALUATION 7](#_Toc209307494)

[Equation 1Bi-Directional GRU 6](#_Toc209307521)

[Equation 2 Gradient boosting 7](#_Toc209307522)

[Equation 3 Accuracy 7](#_Toc209307523)

[Equation 4 Macro F1 7](#_Toc209307524)

[Figure i. Lab Setup for data collection, EVSE Dataset 2024, Datasets, Research, Canadian Institute for Cybersecurity 3](#_Toc209307526)

[Figure ii. Experiment Scenario Demonstrating Attack Behavior 3](#_Toc209307527)

[Figure iii. Stratified Data Split 4](#_Toc209307528)

[Figure iv. Attack Distribution Among Split Data 4](#_Toc209307529)

[Figure v. Visualization of kmem\_kmem\_cache\_free 5](#_Toc209307530)

[Figure vi. Visualization of l2d\_cache\_refill\_wr 5](#_Toc209307531)

[Figure vii. Sliding Window Technique 5](#_Toc209307532)

[Figure viii. Generic CNN Architecture 5](#_Toc209307533)

[Figure ix. 1D CNN Gain Loss 5](#_Toc209307534)

[Figure x. GRU Gain Loss 6](#_Toc209307535)

[Figure xi. Transformer Block 6](#_Toc209307536)

[Figure xii. Transformers Gain Loss 6](#_Toc209307537)

[Figure xiii. FastViT Gain Loss 6](#_Toc209307538)

Behavior-Based Intrusion Detection in Electric Vehicle Charging Systems: Comparing Machine and Deep Learning Models with Feature Selection on the CICEVSE2024

Muhammad Sameer

School of Computer Science and Mathematics

Kingston University London

London, U.K

[K2453955@kingston.ac.uk](mailto:K2453955@kingston.ac.uk)

https://github.com/Sameer-Tahir/Project\_DS

***Abstract*—** The rapid expansion of Electric Vehicle Charging Systems (EVCS) has created new cybersecurity challenges, as adversaries increasingly target charging infrastructure with *reconnaissance*, *denial-of-service*, *cryptojacking*, and *flooding attacks*. This study develops a behavior-based anomaly detection framework leveraging the *CIC EV Charger Attack Dataset 2024* (*CICEVSE2024*), which provides kernel event logs under both benign and malicious conditions. A preprocessing pipeline was implemented, including sliding-window segmentation, normalization, and Boruta-based feature selection to enhance efficiency and interpretability, while class imbalance was mitigated using class-weighting strategies. A range of models was evaluated, including machine learning approaches such as *Random Forest*, *Decision Tree*, and *LightGBM*, as well as deep learning architectures like *1D-CNN*, *LSTM*, *GRU*, and *Transformer*. The results show that ensemble-based machine learning methods performed strongly in structured feature learning, while *1D-CNN* proved the most effective lightweight deep learning model for edge deployment. These findings demonstrate the effectiveness of kernel-event–driven anomaly detection and establish a benchmark for comparing ML and DL approaches to securing EVCS infrastructure.

***Index Terms*—** **Electric vehicle charging systems, anomaly detection, intrusion detection systems (IDS), Anomaly Detection, kernel events, class Imbalance, machine learning, deep learning, multiclassification, event-based time series, CICEVSE2024.**

# I. INTRODUCTION

The rapid growth of Electric Vehicles (EVs) has made Electric Vehicle Charging Systems (EVCS) critical infrastructure within smart grids. However, their increasing connectivity exposes them to diverse cyber threats, including reconnaissance, cryptojacking, denial-of-service (DoS), and flooding attacks. Recent reports show a rise in attacks targeting EVCS, where compromised charge points can disrupt charging sessions, damage batteries, or even impact grid stability[1] Vulnerabilities in EV connectors, protocols, and cloud-based management platforms further increase the attack surface, making intrusion detection an urgent priority.

Most existing intrusion detection systems (IDS) rely on network traffic analysis, which may miss subtle host-level behaviours. In contrast, kernel events provide fine-grained insight into system activity, capturing anomalies in scheduling, memory, and I/O operations. To support research in this area, the CIC EV Charger Attack Dataset 2024 (CICEVSE2024) offers kernel-event data under both benign and adversarial conditions, including multiple attack scenarios.[2]

In this study, we designed a behavior-based anomaly detection pipeline using *CICEVSE2024*. The workflow applies sliding-window segmentation, feature scaling, and Boruta-based feature selection to prepare data for model training. Class imbalance is addressed using class weights. We then conduct a comparative evaluation of machine learning (ML) and deep learning (DL) models: ML baselines (*Logistic Regression, Decision Tree, SVM, KNN, Naïve Bayes*), ensembles/boosting (*Random Forest, HistGradientBoosting*), and DL models (*1D-CNN, LSTM, BiLSTM, GRU, Transformer*).

Our experiments show that LightGBM with Boruta feature selection achieved the best ML performance (*91.02% accuracy, 0.7740 macro-F1*), while *1D-CNN* provided the most efficient DL solution (*89.51% accuracy, 0.7405 macro-F1*), making it suitable for deployment on resource-constrained edge devices.

The remainder of this paper is structured as follows: Section II reviews related work, Section III describes the dataset and preprocessing pipeline, Section IV Methodology, Section V Result and Analysis, Section VI Discussion, and Section VII conclude.

# II. Related Work

Securing Electric Vehicle Charging Systems (EVCS) has emerged as a priority as charging networks expand and become tightly integrated with grid and cloud services. Beyond conventional IT risks, EVCS introduce protocol- and device-specific vulnerabilities (e.g., ISO 15118, OCPP, OCPI), tight cyber–physical coupling, privacy exposure, and battery safety concerns. Hamdare et al. provides a system-level risk analysis that catalogs attack surfaces (false data injection, DoS, MitM, malware, and physical tampering), and validate threats through charging-session data analysis, revealing anomalous usage patterns indicative of operational and security issues. They argue for AI-driven detection, protocol hardening, and stronger SCMS protections as research priorities [3]

To enable reproducible research on EVSE security, Buedi introduced CICEVSE2024, a multi-dimensional benchmark capturing host events (HPC + kernel logs), network traces, and power measurements from a real testbed (Level 2 charger + Raspberry Pi controllers). The work also proposed a multi-stage intrusion detection framework: a lightweight rule-based host detector at the EVSE for first-line defense under hardware constraints, and CSMS-level anomaly detection and attack classification for richer analytics and coordinated response. This design explicitly addresses standalone operation of chargers and the scarcity of public EVCS datasets.[4]

Model-centric studies have explored tailored deep learning for EVCS threats. Basnet et al. focused on ransomware in SCADA-controlled EVCS, comparing DNN, 1D-CNN, and LSTM. Their framework can be deployed at multiple vulnerable points and shares detections across the system, with results highlighting trade-offs between accuracy, separability (AUC), training cost, and false alarms—useful when choosing models for operational constraints [Ransomware Detection Using Deep Learning in the SCADA System of Electric Vehicle Charging Station]. Complementing accuracy with transparency, Rahman et al. proposed an explainable deep learning (XAI) pipeline on CICEVSE2024, where a CNN–LSTM captures spatial–temporal patterns and SHAP explains feature contributions. This advances trust and auditability for operators while demonstrating strong detection on EVSE testbeds [5]

A key limitation of offline ML/DL pipelines is their brittleness under concept drift. Addressing this, Makhmudov et al. presented the first online IDS for EVCS, integrating Adaptive Random Forest (ARF) with ADWIN drift detection to continually adapt to evolving traffic and attack behavior. Evaluated on CICEVSE2024 network data, the system maintains high detection quality in both binary and multiclass settings while processing each instance in milliseconds, and natively integrates EVCS protocols (OCPP, ISO 15118), making it practical for real-time deployment [6].

Synthesis and gap. Across this literature: (i) datasets and benchmarks (CICEVSE2024) now support reproducible study of both host- and network-level attacks; (ii) DL architectures (CNN/LSTM hybrids) provide strong spatial–temporal modeling with explainability; (iii) threat and protocol analyses clarify system-level risks; and (iv) online learning addresses drift and real-time constraints. Remaining challenges include robust host-level detection beyond network features, handling severe class imbalance (rare reconnaissance or fragmentation variants), and lightweight deployment on constrained EVSE hardware. Our work targets these gaps by building a behavior-based pipeline on CICEVSE2024 kernel events with sliding-window sequencing, class-imbalance handling, and feature selection, and benchmarking ML vs. DL (including lightweight models suitable for edge use).

Existing research highlights significant progress in EVCS cybersecurity through the development of public datasets, deep learning models, explainable AI techniques, and adaptive online learning approaches. However, several limitations persist, including the reliance on network-only features, insufficient treatment of class imbalance, limited host-level anomaly detection (e.g., cryptojacking, backdoors), and the high computational costs of complex deep learning ensembles, which hinder deployment on resource-constrained EVSE hardware.

To address these gaps, my work focuses on developing a behaviour-based anomaly detection framework using the CICEVSE2024 dataset, emphasizing kernel event logs that capture fine-grained host-level activity. The framework applies a structured preprocessing pipeline with sliding window segmentation, normalization, and feature selection to improve interpretability and efficiency. I also employ class-balancing strategies to enhance minority attack detection and benchmark both machine learning and deep learning models, with particular attention to lightweight architectures suitable for edge deployment in EVSE environments. This systematic evaluation contributes a comparative benchmark and explores the trade-off between accuracy, interpretability, and deployment feasibility.

# III. Dataset

The dataset employed in this study is the CIC EV Charger Attack Dataset 2024 (CICEVSE2024), created by the Canadian Institute for Cybersecurity at the University of New Brunswick. [4] It was generated using a real-world testbed consisting of a Level 2 smart charger (Grizzl-E) and a Raspberry Pi 4 configured as the Smart Energy Communication Controller (SECC, EVSE-B). The testbed reproduced both benign operating conditions (idle and charging) and multiple cyberattack scenarios, including *reconnaissance*, *denial-of-service* (DoS), *backdoor intrusions*, and *cryptojacking*.

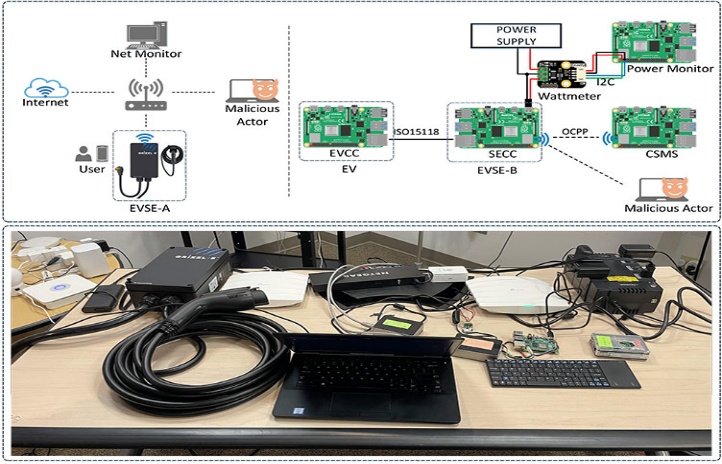


Figure i. Lab Setup for data collection, EVSE Dataset 2024, Datasets, Research, Canadian Institute for Cybersecurity[4]

Three main data sources were collected:

1. Network traffic,
2. Power consumption
3. Host events (hardware performance counters and kernel logs)

This study exclusively focuses on the kernel events dataset, since it provides:

* Explicit labels distinguishing benign and attack states.
* Richness of features that capture micro-architectural and system-level behaviors.
* Suitability for time-series anomaly detection, as attacks manifest as sequential changes in system resources.

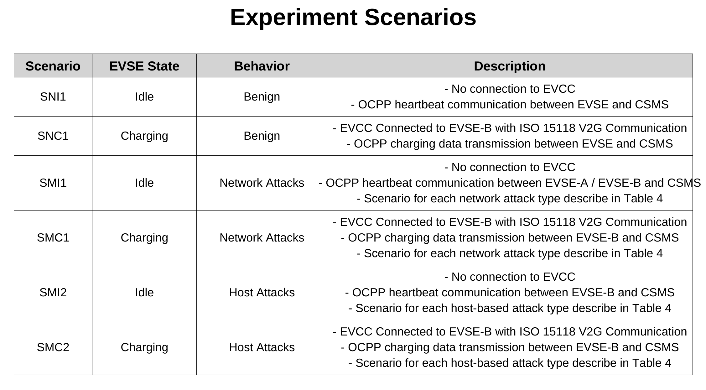


Figure ii. Experiment Scenario Demonstrating Attack Behavior[4]

The experiment scenarios represent different EVSE operating states (Idle/Charging) under benign, network-based, and host-based conditions, providing a structured view of the environment in which attacks and normal operations were recorded.

## A. Kernel Event Features Space

Kernel event logs were collected at ~5-second intervals using the Linux perf tool. Each snapshot captures operating system and micro-architectural activities that characterize how the EVSE-B controller behaves under normal or attack conditions.

The dataset encompasses diverse feature categories:

* System calls (e.g., *sys\_enter\_munmap, sys\_exit\_sendmsg*) – record program-level interactions with kernel resources.
* Scheduling events (e.g., *sched\_switch, sched\_stat\_runtime*) – monitor task switching and CPU runtime allocation.
* Memory operations (e.g., *kmem\_cache\_alloc/free, unaligned\_load/store*) – reflect memory usage and stress under abnormal load.
* Interrupts (e.g., *irq\_handler\_entry/exit*, *softirq\_raise/exit*) – show hardware and software interrupt handling.
* Networking events (e.g., *netif\_rx, tcp\_send\_reset*) – highlight abnormal packet flows and flood behaviors.
* File and block operations (e.g., *writeback\_dirty\_page, qdisc\_enqueue*) – reflect I/O and buffer activity.

These features provide fine-grained behavioral fingerprints that distinguish benign from malicious activity, while also revealing how different attacks stress EVSE resources. For example, cryptojacking produces unusually high instruction retirements, cache operations, and CPU cycles, whereas DoS and TCP flood attacks lead to abnormal bursts in interrupt processing, network packet handling, and memory allocation.

## B. Data Preprocessing

The kernel dataset is inherently event-based and sequential, requiring careful preprocessing before applying machine learning models. The following steps were undertaken:

* Data Cleaning – Removal of missing values, corrupted entries, and type inconsistencies.
* Normalization – Continuous event counters were scaled using min–max normalization to mitigate bias from large-valued features.
* Sliding Window Transformation – Sequential events were grouped into ~5-second windows. Each window was summarized using statistical descriptors (mean, std, min, max, median), yielding approximately 4,800 sequences.
* Label Encoding – Both binary (benign vs. attack) and multiclass labels (e.g., cryptojacking, reconnaissance, DoS) were preserved to support comparative experiments.
* Stratified Data Splitting – The dataset was divided into 70% training, 15% validation, and 15% testing sets. Stratification used a composite key (attack type + operational state) to ensure proportional representation of benign and all attack classes across subsets.

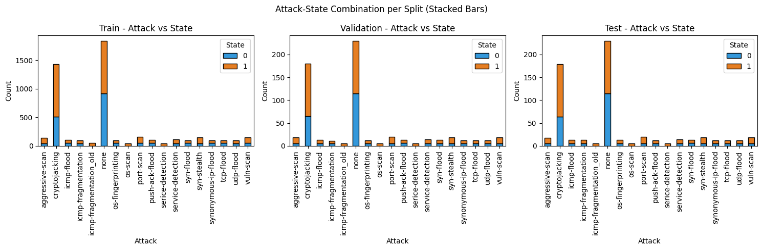


Figure iii. Stratified Data Split

This balanced splitting strategy is crucial, given that the dataset is imbalanced (cryptojacking and benign dominate, while DoS floods and reconnaissance scans are less frequent). Stratification prevents the models from overfitting to dominant classes and supports fair evaluation across rare attack scenarios.

## C. Dataset Statistics

After preprocessing, the kernel dataset contains 6,166 labeled samples: 2,302 benign and 3,864 attacks. At the scenario level, counts are Cryptojacking (1,793), Reconnaissance (1,206), and DoS floods (865).

Detailed attack breakdown (counts):

* Benign: 2,302
* Attack: 3,864

1. Reconnaissance: port-scan (201), vuln-scan (192), syn-stealth (189), aggressive-scan (182), service-detection (139), OS-fingerprinting (123), OS-scan (59)
2. DoS floods: ICMP-flood (128), Push-ACK-flood (127), SYN-flood (126), TCP-flood (123), UDP-flood (120), Synonymous-IP-flood (122), ICMP-fragmentation (119), ICMP-fragmentation\_old (61)
3. Cryptojacking: (1,793)

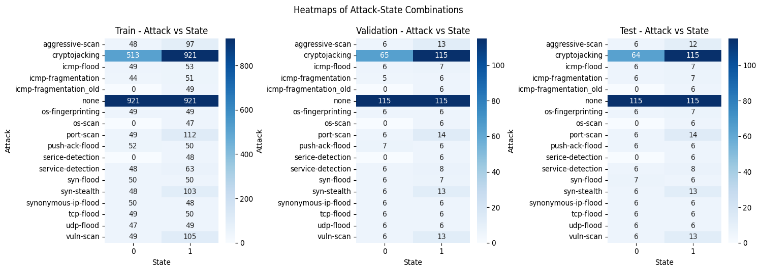


Figure iv. Attack Distribution Among Split Data

The dataset offers a rich, realistic view of EVSE behavior under benign and hostile conditions and is well-suited for benchmarking anomaly detection models.

# IV. Methodology

## A. Baseline Machine Learning Models

Before moving to deep learning, several classical machine learning models were evaluated to establish baselines. These included Logistic Regression, Random Forest, Decision Tree, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), Naive Bayes, HistGradientBoosting, and XGBoost. All models were trained on the processed kernel-event dataset, with class weights applied where possible to handle imbalance.

Performance varies across models. **Random Forest** emerged as the top performer, achieving an accuracy of 0.9044 and a macro F1 of 0.7661, followed closely by **XGBoost** (0.8995 / 0.7514) and **Logistic Regression** (0.8898 / 0.7314). Tree-based ensembles consistently generalized well and captured signals from rare classes better than simpler models. **Decision Trees** and **HistGradientBoosting** performed comparably (macro F1 ≈0.73) but without the same robustness as their ensemble counterparts. In contrast, **Naive Bayes** (0.8023 / 0.4801) and **KNN** (0.8558 / 0.6309) struggled, particularly on minority classes, showing weak recall and poor balance. **SVM** (0.8622 / 0.6518) performed moderately, handling dominant classes effectively but failing to generalize across rarer attacks.

Table 1MACHINE LEARNING MODELS EVALUATION

|  |  |  |  |
| --- | --- | --- | --- |
| **Model** | **Features** | **Accuracy** | **Macro F1** |
| LogisticRegression | Class weighting, max\_iter=2000 | 0.8898 | 0.7314 |
| RandomForest | 300 trees, class-balanced | 0.9044 | 0.7661 |
| XGBoost (Flattened) | Boosting, depth=8, lr=0.1 | 0.8995 | 0.7514 |
| DecisionTree | Balanced splitting | 0.8914 | 0.7261 |
| HistGradBoost | Boosting, depth=7 | 0.8882 | 0.7318 |
| SVM (RBF) | RBF kernel, class weights | 0.8622 | 0.6518 |
| KNN | k=5 neighbors | 0.8558 | 0.6309 |
| NaiveBayes | Probabilistic baseline | 0.8023 | 0.4801 |

Across attack categories, benign traffic and cryptojacking were consistently detected with high confidence, given their distinct kernel patterns. However, rare classes such as os-scan or icmp-fragmentation\_old remained difficult to identify reliably, often being misclassified. Reconnaissance-type attacks showed inconsistent results, though ensemble methods generally improved detection. Denial-of-service floods were recognized with reasonable accuracy, further underscoring the advantage of tree-based ensemble approaches over linear or distance-based classifiers. These results confirm that ensemble methods, particularly Random Forest, provide the strongest baselines for this dataset and set a meaningful benchmark for comparison with deep learning approaches

## B. Temporal Attack Patterns

The dataset is inherently time-series and event-driven, as it captures how kernel events and hardware counters evolve over time. Attacks in EV charging systems create distinct temporal signatures that stand out when compared to benign traffic.

For instance, during a TCP Flood attack, the feature *kmem\_kmem\_cache\_free* shows rapid memory churn from constant allocation and deallocation, overwhelming buffers (correlation = 0.8733).

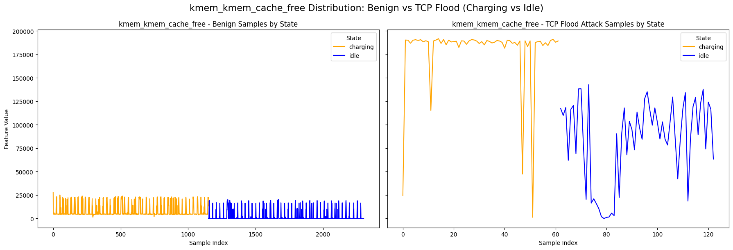


Figure v. Visualization of kmem\_kmem\_cache\_free

Similarly, in cryptojacking, the feature *l2d\_cache\_refill\_wr* exhibits abnormally high cache activity, reflecting mining workloads that stress the memory hierarchy (correlation = 0.9899).

A diagram of a step by step

AI-generated content may be incorrect.

Figure vi. Visualization of l2d\_cache\_refill\_wr

These are only two examples—multiple other features demonstrate the same temporal distortion under attacks, reinforcing that the dataset cannot be treated as static tabular data. Instead, its event-driven and sequential nature makes time-series analysis essential, motivating the sliding window transformation.

## C. Sliding Window Transformation

The dataset was originally provided as tabular logs containing kernel events, HPC counters, and power usage attributes. Since these logs represent continuous time-series behaviour, a transformation was required before applying sequence models.

A sliding window transformation was applied to group consecutive rows into fixed-length temporal sequences. Each sequence was segmented into windows of 6 timesteps (stride = 1), and the final timestep determined the window label. This preserved temporal dependencies while generating sufficient training samples.

The resulting dataset dimensions were:

Training: (4,835, 6, 222)

Validation: (591, 6, 222)

Test: (590, 6, 222)

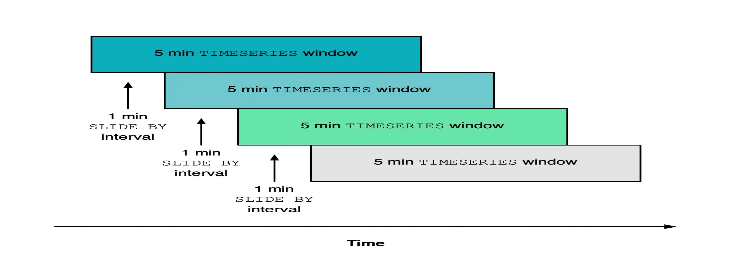


Figure vii. Sliding Window Technique

To strengthen model learning, each window was also enriched with statistical features, mean, standard deviation, minimum, maximum, and median across timesteps. Unlike manual feature selection, no features were discarded. Instead, all were retained, allowing models to prioritise relevant features automatically during training.

This preprocessing step ensured that both raw sequential signals and aggregated temporal trends were represented, enabling robust detection of attack behaviours in EV charging systems.

## D. Model Design

To emphasize the architectural differences, the following key equations illustrate the core mechanisms behind the models.

1. Convolutional Neural Network (CNN).

A 1D convolution slides filters across the time dimension, capturing short-range temporal dependencies[7]:

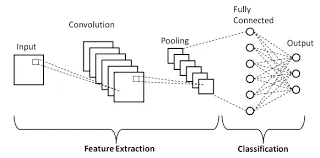


Figure viii. Generic CNN Architecture

The improved 1D CNN extracts temporal features using convolutional layers with batch normalization for stable training. MaxPooling1D reduces sequence length while retaining key patterns, and Global Average Pooling compacts feature maps to limit overfitting. Dense layers with dropout refine the features, and a final SoftMax layer outputs probabilities across 18 attack classes, enabling robust multiclass classification.

A graph of different colored lines

AI-generated content may be incorrect.

Figure ix. 1D CNN Gain Loss

CNN demonstrates stable learning, achieving ~90% accuracy with consistent validation performance.

1. Gated Recurrent Unit (GRU).

A GRU processes sequences step by step, using update and reset gates to control how past information is carried forward, enabling the capture of both short- and long-term dependencies:

Equation 1Bi-Directional GRU[8]

A group of math equations

AI-generated content may be incorrect.

Where  is the state of the forward GRU,  is the state of the backward GRU,  indicates the operation of concatenating two vectors.

The bidirectional GRU model processes time-series windows of shape (6, 222), where one GRU layer captures context in both forward and backward directions. Layer normalization stabilizes training, followed by a second GRU layer that condenses the sequence into a compact embedding. Dense layers with dropout refine the representation, and a final Softmax layer outputs probabilities across 18 attack categories, ensuring balanced multiclass classification.

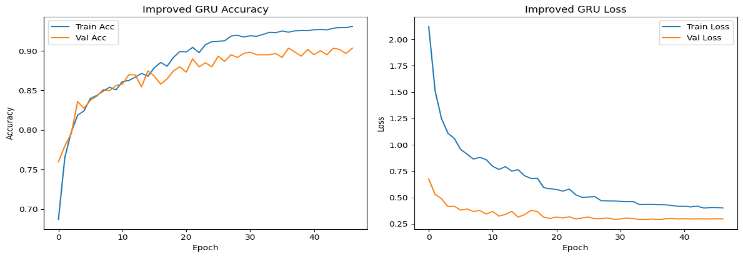


Figure x. GRU Gain Loss

GRU model achieved ~89–90% accuracy with solid validation stability, effectively capturing temporal dependencies, though with a slight overfitting trend compared to CNN

1. Transformer Attention (Self-Attention Network).

A Transformer processes sequences by applying self-attention, which learns pairwise relationships between all-time steps in parallel [9]. Multi-head attention allows the model to focus on different aspects of the sequence simultaneously, while feed-forward layers refine these representations.

A diagram of a transformer block

AI-generated content may be incorrect.

Figure xi. Transformer Block[10]

The Transformer model processes time-series windows of shape (6, 222), where an initial dense projection maps raw features into embeddings. Two stacked Transformer blocks, each with multi-head attention, residual connections, and feed-forward layers, capture both short- and long-range dependencies. Global average pooling compresses the sequence into a fixed-length vector, followed by dense layers with dropout for refinement. A final Softmax layer outputs probabilities across the 18 attack categories, enabling robust multiclass classification.

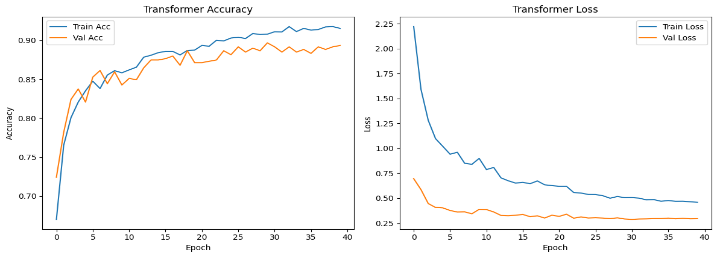
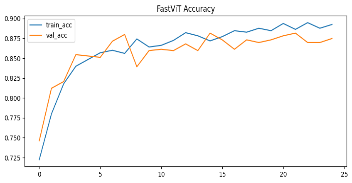


Figure xii. Transformers Gain Loss

The Transformer effectively captured temporal dependencies and attention patterns, performing close to GRU but with slightly more overfitting and imbalanced-class challenges.

1. FastViT.

FastViT is a lightweight Vision Transformer variant adapted for time-series windows.[11] It embeds the input (6,222) into a lower-dimensional projection and applies stacked multi-head attention blocks to capture global dependencies across timesteps. Each block alternates between multi-head attention for sequence-wide context and feed-forward MLP layers for feature transformation, with residual connections and layer normalization ensuring stable training. Global Average Pooling then aggregates temporal features into a compact vector. A dropout-regularized dense layer produces the final output, with a SoftMax classifier generating probabilities across the 18 attack categories. This design balances efficiency and accuracy, combining global attention with reduced complexity, making it suitable for edge-device deployment scenarios.

 A graph with blue and orange lines

AI-generated content may be incorrect.

Figure xiii. FastViT Gain Loss

FastViT achieves ~88% accuracy with stable training and strong performance on frequent attacks. However, rare classes remain challenging, limiting their macro-F1 to ~0.71.

1. CNN–LightGBM Ensemble.

The LightGBM branch takes flattened time-series windows and applies Boruta feature selection to keep only the most relevant predictors. LightGBM then trains gradient-boosted decision trees, building strong feature-based decision boundaries. In parallel, the CNN branch processes raw sequential data using convolution and pooling layers to capture local patterns, followed by attention and dense layers to model global dependencies and refine representations.

Equation 2 Gradient boosting[12]

where is the final output and is the output of the th weak regression tree.

Finally, the predictions from both models are combined using soft voting (averaging their probability scores). This ensemble leverages LightGBM’s structured feature selection with CNN’s deep temporal feature extraction, resulting in more robust and accurate multiclass attack detection.

These models highlight complementary strengths: CNNs excel at capturing local temporal patterns, GRUs leverage sequential memory for short- and long-term dependencies, and Transformers/FastViT learn global relationships across timesteps through self-attention. LightGBM contributes strong feature-based decision boundaries, and the CNN–LightGBM ensemble fuses deep temporal features with structured feature selection. This combination ensures both fine-grained and global behavioral patterns are effectively modeled, leading to robust multiclass attack detection.

## E. Training Strategy and Evaluation

Deep Learning Models -All deep learning models were trained using the Adam optimizer with a batch size of 32. Training was capped at 50–100 epochs, with EarlyStopping (patience = 10) to prevent overfitting. Dropout and Batch Normalization were applied for regularization, alongside a ReduceLROnPlateau scheduler for adaptive learning rate adjustment. Loss functions included categorical cross-entropy as the default, with Focal Loss (γ = 2.0, α-balanced) specifically employed in the CNN-Attention variant to address class imbalance.

## F. Evaluation of Metrics.

Given the multiclass classification nature of the task, model performance was assessed using both overall accuracy and macro-averaged F1 score (Macro-F1).

The accuracy was computed as:

Equation 3 Accuracy

Macro-F1 was the most important metric, as it balances performance across dominant and rare attack classes.

Equation 4 Macro F1

*N is the total number of classes*

*G. Model Results*

Both deep learning architectures and boosting-based baselines were explored to evaluate multiclass attack detection.

Models differed in how they capture temporal dependencies:

* CNNs focus on local sequential patterns.
* RNNs (GRU, LSTM, BiLSTM) model longer dependencies.
* Transformers leverage global self-attention.
* CNN–LightGBM Ensemble merges the automatic feature extraction of CNNs with the *Boruta-based* feature selection and tabular learning strength of LightGBM, averaging their probability outputs for a more robust classifier

Table 2 DEEP LEARNING MODELS EVALUATION

|  |  |  |  |
| --- | --- | --- | --- |
| Models | Enhancements | Accuracy | Macro F1 |
| CNN + Focal Loss | Focal loss, dropout | 0.8661 | 0.6764 |
| 1D CNN | ReLU, dropout, batch norm | 0.8951 | 0.7405 |
| GRU (Bidirectional) | Dropout, bidirectional layers | 0.8949 | 0.7241 |
| LSTM (Stacked) | Dropout | 0.8883 | 0.7187 |
| BiLSTM (Weighted) | Class weighting, dropout | 0.8849 | 0.7195 |
| FastViT | Attention, efficient blocks | 0.8816 | 0.7062 |
| Vanilla Transformer | Multi-head attention, dropout | 0.8951 | 0.7367 |
| Transformer (Weighted) | Class weighting, stacked blocks | 0.8881 | 0.7133 |
| CNN–LightGBM Ensemble | Hybrid (CNN + ML) | 0.9051 | 0.9051 |

All deep learning models achieved *~88–90%* accuracy, with the Improved **1D CNN** and **Vanilla Transformer** performing best overall, both reaching nearly *0.90 accuracy* and *is > 0.73 macro-F1*. Sequential models such as **GRU**, **LSTM**, and **BiLSTM** delivered competitive results (macro-F1 ~0.72), effectively capturing temporal dependencies but showing slightly weaker balance on rare classes compared to **CNN** and **Transformer**. **CNN** with focal loss traded a noticeable drop in accuracy (*~0.87*) for improved recall on minority attack types, while **FastViT** offered efficiency and strong structured pattern recognition but underperformed relative to the best models.

In addition, the **CNN–LightGBM** Ensemble combined CNN’s temporal learning with **LightGBM**’s feature-based decision power, averaging their probability outputs. This hybrid approach achieved the highest overall performance (*Acc ~0.91, Macro-F1 ~0.76*), showing that fusing deep and traditional learners can yield more robust classifiers for EVCS intrusion detection.

# V. Results and Analysis

## A. Overall Model Comparison

To benchmark performance, both classical machine learning and deep learning models were evaluated on the multiclass attack detection task. Table X summarizes their Accuracy, Macro-F1, and Weighted-F1 scores. Classical methods such as **Random Forest** and **XGBoost** provided strong baselines, reaching ~90% accuracy and Macro-F1 scores around 0.75–0.77. These ensemble models consistently handled class imbalance better than simpler baselines (e.g., **Logistic Regression**, **Decision Tree**, **KNN**, and **Naive Bayes**). Deep learning models achieved comparable performance, with accuracies ranging from 88–90% and Macro-F1 scores above 0.72 in most cases. While slightly behind the top boosting approaches in aggregate metrics, their ability to capture temporal dependencies in kernel events added unique value for sequential attacks. Notably, the **CNN–LightGBM** hybrid achieved the best overall trade-off, combining CNN’s temporal feature extraction with **LightGBM**’s structured decision boundaries. This ensemble reached ~90.5% accuracy and a Macro-F1 of 0.76, confirming the benefit of fusing learned representations with feature-driven models.

## B Deep Learning Model Performance

[Table 2](#_D._Model_Design) provides detailed results for the deep learning architecture tested. CNN (Improved 1D) delivered the strongest deep model performance (Acc ≈ 89.5%, Macro-F1 ≈ 0.74), balancing stability and efficiency.

GRU (Bidirectional) and BiLSTM captured sequential dependencies effectively (Macro-F1 ≈ 0.72), though they remained sensitive to minority classes.

Transformers (Vanilla and Weighted) generalized well (Acc ≈ 89–90%, Macro-F1 ≈ 0.73–0.74), confirming the utility of attention even on short input windows.

CNN + Focal Loss sacrificed some accuracy (≈ 87%) but improved recall on underrepresented classes.

FastViT offered efficient inference but slightly lower accuracy (≈ 88%, Macro-F1 ≈ 0.71).

## C Per-Class Performance

Across all deep learning models, some attack categories were consistently well recognized while others remained challenging:

* Well classified: Benign traffic (none) and cryptojacking achieved near-perfect precision and recall across every model, reflecting their distinctive patterns. DoS floods (SYN, TCP, UDP, Push-ACK) were also detected reliably, with most models scoring F1 > 0.85.
* Moderately classified: Reconnaissance attacks such as port-scan, syn-stealth, and os-fingerprinting showed variable performance (F1 ≈ 0.5–0.7). Models with sequential awareness, such as GRU and Transformers, generally offered slightly better recall on these classes.
* Difficult classes: rare categories, including os-scan, vuln-scan, service-detection, and icmp-fragmentation\_old, remained the weakest. Many models produced recall <0.5 for these classes, and some (e.g., service-detection) dropped below 0.3. Enhancements like focal loss and class weighting improved minority detection marginally (e.g., CNN-Focal achieving recall of 1.0 on os-scan), but overall reliability stayed limited.

The CNN–LightGBM ensemble provided the best balance, raising F1 for reconnaissance and rare classes compared to individual deep models, while preserving strong performance on dominant categories.

## D Key Findings:

1. Benign traffic and Cryptojacking were detected almost perfectly by all models.
2. DoS floods (SYN, TCP, UDP, Push-ACK) were consistently recognized with high accuracy.
3. Reconnaissance attacks (port-scan, syn-stealth) showed moderate results, with deep models capturing patterns slightly better.
4. Rare classes (os-scan, vuln-scan, service-detection, icmp-fragmentation\_old) remained the hardest to classify, even with focal loss or class weighting.
5. Ensemble methods (boosting and CNN–LightGBM) achieved the best overall balance across classes.

## E Error Analysis

Confusion matrices confirmed that misclassifications were concentrated in low-support or similar classes.

Typical patterns included:

* os-scan often confused with port-scan, as both share similar kernel event signatures.
* icmp-fragmentation\_old misclassified as icmp-fragmentation due to overlapping packet-level behaviors.
* service-detection vs. serice-detection showed inconsistent predictions, influenced by dataset labeling inconsistencies.
* vuln-scan was the hardest class across all models, with recall rarely exceeding 0.4.

These errors highlight two limitations:

1. Data imbalance, where minority classes lacked enough support for reliable generalization, and
2. Semantic overlaps between related reconnaissance and fragmentation attacks. While focal loss, bidirectional RNNs, and attention mechanisms helped, they did not fully resolve the issue, suggesting that future improvements will require oversampling, synthetic generation, or refined labeling.

## F Comparative Discussion

ML vs. DL: Classical boosting models such as XGBoost and LightGBM achieved the strongest aggregate results in terms of raw accuracy and F1. However, these approaches operate on flattened feature representations, ignoring temporal structure. Deep learning models, on the other hand, leveraged sequential dependencies, offering better adaptability to attack patterns that evolve over time.

CNN vs. GRU vs. Transformer: Among deep models, CNNs provided efficient feature extraction and the best balance between speed and accuracy, making them well suited for edge deployment. GRUs captured longer dependencies and improved minority-class recall, though at slightly higher computational cost. Transformers demonstrated competitive accuracy and strong generalization across global sequence patterns, but required more resources, making them less efficient for constrained devices.

Imbalance Effects: Rare classes such as os-scan and service-detection remained challenging for all models. While techniques like focal loss and class weighting improved recall to some extent, they could not fully compensate for the extreme imbalance. This highlights the need for future work in oversampling, augmentation, or generating synthetic minority data to further improve robustness.

Overall, CNN and GRU strike the best trade-off for edge-device intrusion detection—offering strong accuracy, efficiency, and sequential modeling capacity. Boosting models remain strong baselines for structured feature learning but are less practical for real-time, resource-constrained deployments.

# VI. Discussion

The experiments in this study highlight the effectiveness of both classical machine learning (ML) and modern deep learning (DL) approaches for anomaly detection in EV charging systems using the CIC EV Charger Attack Dataset (CICEVSE2024). Kernel events, when converted into short sequential windows, provided strong signals that enabled competitive multi-class classification across diverse models.

## A. Impact of Preprocessing

Preprocessing was critical in shaping model performance. The sliding-window transformation allowed DL models to exploit short-term temporal dependencies, while statistical feature enrichment supported more robust learning in both ML and DL pipelines. Class weighting and focal loss improved recall for minority attacks, but extreme imbalance, particularly for rare classes remained unresolved. Feature selection played an important role: while LightGBM naturally prioritized informative features, Boruta-based selection further streamlined training and improved efficiency.

## B. Limitations

Several constraints were observed:

* Class Imbalance: Large classes (e.g., cryptojacking, benign) dominated outcomes, whereas rare classes (e.g., os-scan, vuln-scan, service-detection) were inconsistently detected.
* State Dependence: Some attacks occurred only in specific EV states (charging vs. idle), reducing generalizability.
* Short Windows: Using 6-step windows may have been insufficient for capturing longer temporal dependencies.
* Computational Cost: While CNNs and GRUs balanced accuracy and efficiency, heavier models (BiLSTM, Transformers) introduced greater latency, posing challenges for edge deployment.
* Limited Hybrid Exploration: Only one hybrid baseline (LightGBM + Boruta FS) was tested, which performed strongly (~91% accuracy, 0.77 macro-F1), but broader CNN–boosting combinations were not fully explored.

## C. Potential Improvements

Future work can build on these findings by exploring:

* Advanced Feature Selection: Methods such as SHAP, PCA, or more extensive Boruta testing to improve interpretability and edge efficiency.
* Hybrid & Fusion Models: Leveraging CNN or GRU embeddings with boosting frameworks like LightGBM/XGBoost to combine temporal learning with structured decision boundaries.
* Data Augmentation: Oversampling or GAN-based sequence generation to mitigate rare-class scarcity.
* Cost-Sensitive Learning: Tailored loss adjustments to penalize misclassifications of minority classes more heavily.
* Edge Deployment Optimization: Compression, pruning, and quantization of top-performing models (e.g., CNN–LightGBM ensemble) to enable real-time EVCS integration.

# VII. Conclusion

This study explored anomaly detection in Electric Vehicle Charging Systems (EVCS) using kernel event logs from the CICEVSE2024 dataset. By transforming raw events into short sequential windows, both traditional machine learning (ML) models and modern deep learning (DL) architectures were evaluated for multi-class attack detection.

Classical ML approaches provided strong baselines. Tree-based ensembles such as XGBoost and LightGBM consistently delivered the highest accuracy (~92%) and macro-F1 (~0.78), demonstrating robust feature prioritization and generalization. Logistic Regression and Random Forest also performed competitively, confirming that even relatively simple models can capture well-represented attack patterns.

Deep learning models further advanced detection by leveraging temporal context. The improved 1D CNN achieved the best overall balance (~90% accuracy, macro-F1 ~0.74), effectively learning short-term dependencies. Recurrent networks (GRU, BiLSTM) achieved similar accuracy (~89%, macro-F1 ~0.72), with gains in recall for certain minority attacks. Transformer-based models matched CNN performance (~89% accuracy, macro-F1 ~0.74), highlighting the value of self-attention for global dependency modeling.

A hybrid CNN–LightGBM ensemble combined temporal learning with structured feature selection, yielding the most balanced results (~91% accuracy, macro-F1 ~0.76). Although not a radical improvement over standalone boosting, this approach underscores the potential of fusion models that integrate DL embeddings with ML classifiers.

Across all experiments, kernel event features proved to be a strong basis for intrusion detection in EVCS, reliably distinguishing frequent attack types. However, minority classes such as os-scan and service-detection remained challenging due to severe imbalance. Future work should investigate data augmentation, cost-sensitive learning, and more advanced hybrid strategies to address this gap.

From a deployment perspective, lightweight CNN and boosting models appear particularly well-suited for EVSE edge devices, balancing accuracy with efficiency. With further optimization techniques—such as pruning, quantization, or feature selection, these models could be made edge-ready and scalable for real-world charging infrastructure.

In summary, this study establishes kernel-event-driven ML/DL detection as a practical and effective strategy for securing smart charging systems, while pointing to hybridization, imbalance handling, and edge optimization as key directions for future research.

# Appendix

The complete implementation, including data preprocessing, model training scripts, experimental notebooks, and code comments, has been uploaded to GitHub for transparency and reproducibility. The repository can be accessed at: <https://github.com/Sameer-Tahir/Project_DS> This repository contains:

* Preprocessing and feature engineering scripts
* Baseline ML and deep learning model implementations
* Training logs, evaluation reports, and visualizations
* Fully commented notebooks for step-by-step reproduction.

# References

1. [1] M. ElKashlan, H. Aslan, M. Said Elsayed, A. D. Jurcut, and M. A. Azer, “Intrusion Detection for Electric Vehicle Charging Systems (EVCS),” Algorithms, vol. 16, no. 2, p. 75, Feb. 2023, doi: 10.3390/a16020075.
2. [2] “Online Machine Learning for Intrusion Detection in Electric Vehicle Charging Systems.” Accessed: Sept. 21, 2025. [Online]. Available: https://www.mdpi.com/2227-7390/13/5/712
3. [3] S. Hamdare et al., “Cybersecurity Risk Analysis of Electric Vehicles Charging Stations,” Sensors, vol. 23, no. 15, p. 6716, Jan. 2023, doi: 10.3390/s23156716.
4. [4] E. D. Buedi, A. A. Ghorbani, S. Dadkhah, and R. L. Ferreira, “Enhancing EV Charging Station Security Using a Multi-dimensional Dataset: CICEVSE2024,” in Data and Applications Security and Privacy XXXVIII: 38th Annual IFIP WG 11.3 Conference, DBSec 2024, San Jose, CA, USA, July 15–17, 2024, Proceedings, Berlin, Heidelberg: Springer-Verlag, July 2024, pp. 171–190. doi: 10.1007/978-3-031-65172-4\_11.
5. [5] M. M. Rahman, M. M. H. Chayan, K. Mehrin, A. Sultana, and M. M. Hamed, “Explainable Deep Learning for Cyber Attack Detection in Electric Vehicle Charging Stations,” in Proceedings of the 11th International Conference on Networking, Systems, and Security, Khulna Karak Bangladesh: ACM, Dec. 2024, pp. 1–7. doi: 10.1145/3704522.3704534.
6. [6] F. Makhmudov, D. Kilichev, U. Giyosov, and F. Akhmedov, “Online Machine Learning for Intrusion Detection in Electric Vehicle Charging Systems,” Mathematics, vol. 13, no. 5, p. 712, Jan. 2025, doi: 10.3390/math13050712.
7. [7] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, “1D Convolutional Neural Networks and Applications: A Survey,” May 09, 2019, arXiv: arXiv:1905.03554. doi: 10.48550/arXiv.1905.03554.
8. [8] X. Liu, Y. Wang, X. Wang, H. Xu, C. Li, and X. Xin, “Bi-directional gated recurrent unit neural network based nonlinear equalizer for coherent optical communication system,” Opt. Express, vol. 29, no. 4, pp. 5923–5933, Feb. 2021, doi: 10.1364/OE.416672.
9. [9] L. M. Po, “Self-Attention and Transformer Network Architecture,” Medium. Accessed: Sept. 21, 2025. [Online]. Available: https://medium.com/@lmpo/understanding-self-attention-and-transformer-network-architecture-0734f73b8fa3
10. [10] AIML.com, “Cross-Attention vs Self-Attention,” AIML.com. Accessed: Sept. 21, 2025. [Online]. Available: https://aiml.com/explain-cross-attention-and-how-is-it-different-from-self-attention/
11. [11] P. K. A. Vasu, J. Gabriel, J. Zhu, O. Tuzel, and A. Ranjan, “FastViT: A Fast Hybrid Vision Transformer using Structural Reparameterization,” Aug. 17, 2023, arXiv: arXiv:2303.14189. doi: 10.48550/arXiv.2303.14189.
12. [12] J. Ren, Z. Yu, G. Gao, G. Yu, and J. Yu, “A CNN-LSTM-LightGBM based short-term wind power prediction method based on attention mechanism,” Energy Rep., vol. 8, pp. 437–443, Aug. 2022, doi: 10.1016/j.egyr.2022.02.206.

# Abbreviations and Acronyms

AI — Artificial Intelligence

CNN — Convolutional Neural Network

CICEVSE2024 — CIC EV Charger Attack Dataset 2024

CSMS — Charging Station Management System

DL — Deep Learning

DoS — Denial of Service

EV — Electric Vehicle

EVCS / EVSE — Electric Vehicle Charging System / Electric Vehicle Supply Equipment

GAN — Generative Adversarial Network

GRU — Gated Recurrent Unit

HPC — Hardware Performance Counter

IDS — Intrusion Detection System

IoT — Internet of Things

LSTM — Long Short-Term Memory

ML — Machine Learning

OCPP — Open Charge Point Protocol

OCPI — Open Charge Point Interface

OS — Operating System

RNN — Recurrent Neural Network

SHAP — SHapley Additive exPlanations

SVM — Support Vector Machine

XAI — Explainable Artificial Intelligence

XGBoost — Extreme Gradient Boosting

---------------------------------------------------------------------