An Intelligent Data-Driven Model to Secure Intra-Vehicle Communication Using Machine Learning

D.Rajani kumar[1], Dr. T. Rajani Devi[2], Mariyam Sulthana[3]

[12]Assistant Professor, Department of Computer Science, University College for Women, Warangal, Telangana

[3]Student, Department of Computer Science, University College for Women, Warangal, Telangana

ABSTRACT

The rapid evolution of vehicles into sophisticated cyber-physical systems has introduced critical cybersecurity vulnerabilities, particularly in intra-vehicle communication networks. The Controller Area Network (CAN) bus, the de facto standard for in-vehicle communication, was designed for reliability and real-time performance in an era of isolated mechanical systems, lacking fundamental security measures such as authentication, encryption, or message integrity verification. This design flaw enables attackers who gain network access to inject arbitrary messages, potentially causing life-threatening consequences including brake disablement or steering manipulation. This paper presents an intelligent data-driven intrusion detection system that employs machine learning to secure intra-vehicle communication. The proposed system monitors CAN bus traffic in real-time, learning normal communication patterns and detecting anomalies indicative of cyberattacks. Three distinct machine learning architectures are systematically evaluated: Random Forest for baseline classification with statistical features, Convolutional Neural Network (CNN) for automatic local pattern extraction from message sequences, and Long Short-Term Memory (LSTM) network for modeling long-term temporal dependencies. The models are trained and evaluated on the publicly available 'Car Hacking: Attack & Defense Challenge' dataset, which contains normal driving data and four attack types: Denial of Service (DoS), Fuzzy, RPM Spoofing, and Gear Spoofing. Experimental results demonstrate that the LSTM model achieves the highest overall accuracy of 99.4% with an F1-score of 0.97, excelling particularly in detecting context-dependent spoofing attacks. The CNN model achieves 98.5% accuracy with 0.94 F1-score, providing a strong balance between performance and computational efficiency. The Random Forest model achieves 97.8% accuracy with 0.90 F1-score and sub-millisecond inference time, making it ideal for high-volume attack detection such as DoS (F1=0.99). Per-attack analysis reveals that LSTM significantly outperforms Random Forest on Fuzzy attacks (0.96 vs 0.88) and Gear Spoofing (0.95 vs 0.86), while all models perform exceptionally on DoS attacks. This comprehensive comparative analysis provides a clear roadmap for automotive security architects, suggesting that layered or hybrid detection systems combining multiple models offer the most robust defense-in-depth strategy for next-generation connected vehicles.

Keywords—Automotive Cybersecurity; CAN Bus; Intrusion Detection System; LSTM; CNN; Random Forest; Machine Learning; Intra-Vehicle Communication; Anomaly Detection; CAN Bus Security.

I. INTRODUCTION

The automobile has undergone a fundamental metamorphosis—from a purely mechanical machine to a sophisticated cyber-physical system. Contemporary vehicles house between 70 and 100 Electronic Control Units (ECUs), each governing a specific domain such as engine management, transmission control, advanced driver-assistance systems (ADAS), infotainment, or body electronics [1]. These distributed processors must exchange sensor readings and actuation commands in microseconds to coordinate vehicle functions seamlessly, forming an internal network whose responsiveness is non-negotiable for passenger safety.

The Controller Area Network (CAN) bus, conceived by Robert Bosch GmbH in the 1980s and standardized in ISO 11898, serves as the central nervous system of this intra-vehicle network [13]. Its differential signaling tolerates electrical noise, its priority-based arbitration ensures that safety-critical messages preempt lower-priority traffic, and its broadcast topology eliminates point-to-point wiring complexity. These properties made CAN an engineering triumph for its era. However, the protocol was designed when vehicles were electronically isolated; authentication, message integrity verification, and encryption were considered unnecessary overhead [15], [16].

The cybersecurity landscape has changed irrevocably. Modern vehicles incorporate cellular modems, Bluetooth stacks, Wi-Fi adapters, and V2X transceivers—each representing a gateway that an adversary may exploit to reach the internal CAN bus [14]. Once on the bus, an attacker can broadcast spoofed messages indistinguishable from legitimate ECU traffic, because the CAN standard provides no mechanism to verify message origin. High-profile demonstrations—most notably the 2015 remote compromise of a Jeep Cherokee—proved that such exploits are not theoretical: researchers disabled the transmission and brakes of a moving vehicle via a cellular connection [17].

Machine learning offers a compelling countermeasure. Rather than relying on cryptographic handshakes that would violate real-time constraints, a well-trained model can silently observe the statistical and temporal regularities of normal CAN traffic and raise an alert the instant those regularities are disturbed [2], [3]. This behavioral approach is particularly valuable for detecting zero-day attacks whose signatures are unknown a priori—a limitation shared by every rule-based or signature-matching intrusion detection system.

This paper makes the following contributions: (i) a systematic comparison of three architecturally distinct ML classifiers—Random Forest, CNN, and LSTM—on a standardized benchmark; (ii) a per-attack-type analysis that reveals which model is best suited to which threat category; and (iii) an evaluation of inference latency relative to real-time CAN bus requirements, offering practical guidance for automotive deployment.

II. LITERATURE REVIEW

A. Foundational Work in Automotive Security

Koscher et al. [1] conducted the first systematic empirical analysis of automotive attack surfaces, demonstrating physical access to the OBD-II port could yield complete control of safety-critical functions including brakes and steering. While proposing no mitigation, their work catalyzed an entire research discipline. Checkoway et al. [14] extended this surface to wireless channels, enumerating Bluetooth, cellular, and even tire-pressure sensors as viable entry points. Hoppe, Kiltz, and Dittmann [15] subsequently cataloged practical CAN attack vectors and proposed short-term countermeasures, framing the challenge for subsequent intrusion detection research.

B. Machine Learning Approaches for CAN Intrusion Detection

Taylor, Leblanc, and Japkowicz [2] were among the first to employ deep learning for CAN intrusion detection, training a Recurrent Neural Network to predict the next message ID in a CAN stream and flagging deviations as anomalies. The approach demonstrated the potential of sequence modeling but was limited by the computational cost of RNNs at the time. Hossain et al. [3] refined this direction using LSTM networks on the HCRL Car Hacking dataset, achieving detection accuracy exceeding 99% across multiple

attack types—setting the performance benchmark for subsequent studies.

Song, Woo, and Kim [4] introduced one-dimensional CNNs to the problem, showing that local temporal patterns in CAN message sequences could be captured by convolutional filters without the sequential dependencies required by recurrent models. Their work established CNNs as a computationally efficient alternative, especially for attacks exhibiting localized signatures. Kang and Kang [5] demonstrated that even a classical Random Forest classifier, fed hand-engineered statistical features such as message-ID frequency histograms and payload entropy, could achieve competitive detection rates with sub-millisecond inference—critical for ECUs with limited compute budgets.

Seo, Song, and Kim [6] contributed the HCRL Car Hacking benchmark dataset used in this study, enabling reproducible comparison of methods across DoS, Fuzzy, RPM Spoofing, and Gear Spoofing attack types. Ashraf et al. [7] addressed cross-vehicle generalization through transfer learning, demonstrating that a model pre-trained on one vehicle's CAN data could be fine-tuned for a new platform with minimal additional labeled data. Khan et al. [8] introduced explainability via SHAP-based attribution to autoencoder-based anomaly detection, increasing trust in model decisions—a prerequisite for regulatory acceptance.

C. Comparative Analysis and Research Gaps

Despite this body of work, systematic side-by-side comparisons of traditional and deep learning approaches on identical preprocessing pipelines remain scarce. Per-attack-type performance analysis is even less common, leaving practitioners without guidance on which architecture to deploy against which threat. Table I summarizes key related work and identifies the position of this study within the literature.

TABLE I. Comparative Analysis of Related Work

System / Paper	Technique	Key Strength	Limitation
Koscher et al. (2010)	Physical exploitation	Foundational vulnerability proof	No IDS proposed
Taylor et al. (2016)	RNN sequence prediction	Captures temporal sequences	High compute cost
Hossain et al. (2020)	LSTM networks	Accuracy >99% on benchmark	Single architecture
Song et al. (2021)	1D CNN	Computationally efficient	Misses long-range deps.
Kang & Kang (2016)	Random Forest	Lightweight, interpretable	Manual feature eng.
Seo et al. (2018)	Benchmark dataset	Enables reproducible research	Not a detection model
Ashraf et al. (2022)	Transfer learning	Reduces new-vehicle data need	Needs suitable source
Khan et al. (2023)	Autoencoder / XAI	Detects zero-day attacks	Lower accuracy
Proposed Work	RF + CNN + LSTM	Multi-model comparison	Offline evaluation only

III. SYSTEM ARCHITECTURE AND DESIGN

A. Architecture Overview

The proposed system follows a five-stage machine learning pipeline: (1) Data Ingestion, (2) Preprocessing and Feature Engineering, (3) Model Training, (4) Inference and Detection, and (5) Evaluation and Analysis. This modular design isolates each concern, enabling independent optimization and transparent benchmarking.

The Data Ingestion module reads raw CSV logs from the HCRL dataset and labels each row as benign (0) or malicious (1) while preserving the attack-type identifier for per-attack analysis. The Preprocessing module converts hexadecimal CAN IDs to integers, normalizes payload bytes, constructs sliding-window sequences of length 100 for deep learning models, and extracts statistical feature vectors for the Random Forest. The Model Training module trains each classifier on 80% of the data under a stratified split, with early stopping applied to deep learning models. The Inference module classifies unseen CAN messages in near-real-time. The Evaluation module computes accuracy, precision, recall, F1-score, and inference latency, decomposing results by attack type.

B. Feature Engineering

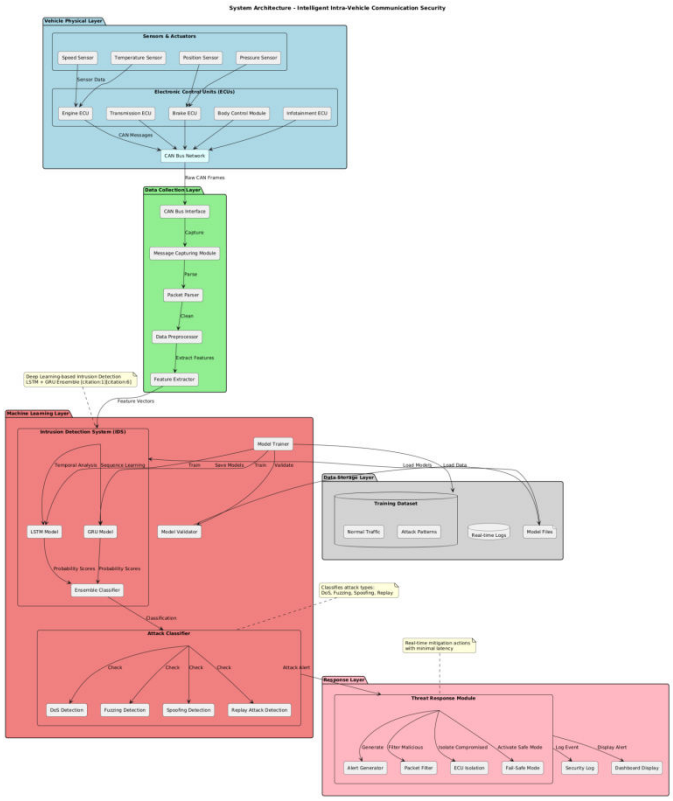
For the Random Forest, a sliding window of W = 100 consecutive messages is characterized by: (i) the count of unique CAN IDs within the window, (ii) per-ID message frequency, (iii) mean and standard deviation of payload byte values, and (iv) payload density (bytes per message). These seven features capture both the statistical distribution and the volumetric anomalies characteristic of DoS and fuzzy attacks.

For CNN and LSTM models, each window is represented as a 3-D tensor of shape (100, 5), where the five features per time step are the integer CAN ID and the first four payload bytes. Normalizing CAN IDs to [0, 1] by dividing by the maximum ID value (0x7FF = 2047) prevents domination of the loss function by large integer values. Payload bytes are divided by 255. The sequence label is the ground-truth label of the final message in the window.

The mathematical expression for sequence normalization is given by:

$$\hat{x}_i = (x_i - x_{min}) / (x_{max} - x_{min})$$

where x_i is the raw feature value, x_{min} and x_{max} are the minimum and maximum values observed in the training set.



C. Model Architectures

Random Forest: An ensemble of 100 decision trees, each grown on a bootstrap sample with Gini impurity as the splitting criterion and a maximum depth of 20. The majority vote of all trees determines the final classification. Ensemble averaging reduces variance without increasing bias, yielding robust generalization to unseen statistical feature combinations.

Convolutional Neural Network: A two-stage 1-D convolutional architecture comprising Conv1D(64, kernel=3, ReLU) → MaxPool(2) → Conv1D(32, kernel=3, ReLU) → MaxPool(2) → Flatten → Dense(50, ReLU) → Dropout(0.3) → Dense(1, sigmoid). Convolutional filters learn local temporal motifs in the message sequence, and max-pooling provides translation invariance.

LSTM Network: A two-layer recurrent architecture: LSTM(64, return_sequences=True) → Dropout(0.2) → LSTM(32) →

Dropout(0.2) → Dense(16, ReLU) → Dense(1, sigmoid). The gating mechanism of the LSTM cell enables selective retention of long-range dependencies across the 100-message window, which is critical for detecting stealthy spoofing attacks whose signature spans many messages.

IV. IMPLEMENTATION

A. Development Environment

All experiments were conducted on a workstation running Windows 11 Pro with an Intel Core i7-12700H (14 cores), 32 GB DDR5 RAM, and an NVIDIA GeForce RTX 3060 GPU (6 GB VRAM). The software stack comprised Python 3.10.11, pandas 1.5.3, NumPy 1.24.3, scikit-learn 1.2.2, TensorFlow 2.13.0, Keras 2.13.1, imbalanced-learn 0.10.1, and matplotlib 3.7.1. Development and training were performed in Jupyter Notebook 6.5.4.

B. Dataset

The HCRL Car Hacking dataset [9] contains CAN traffic logs captured from a KIA Soul under normal driving and four attack scenarios. The normal sub-dataset spans approximately 20 minutes and contains 4,587,542 messages. Each attack sub-dataset spans approximately 5 minutes. Table II presents dataset statistics. The class imbalance (attack messages constituting 7.5–14.6% of each attack dataset) was addressed using SMOTE oversampling on the training split for the Random Forest and class-weight balancing for deep learning models.

C. Training Protocol

Each model was trained using an 80/20 stratified train/test split. The Random Forest required approximately 15 minutes of training. The CNN converged in approximately 45 minutes over 20 epochs with early stopping (patience = 3) monitoring validation loss. The LSTM required approximately 2 hours over 20 epochs. The Adam optimizer with a learning rate of 0.001 was used for both deep learning models; binary cross-entropy served as the loss function. Batch size was set to 256 for memory efficiency.

The sequence creation function is formalized as: given a time-ordered array D of N messages, a sequence $S_i = D[i : i + L]$ where $L = 100$ is the sequence length. The label assigned to S_i is the ground-truth label of $D[i + L - 1]$. This produces $N - L + 1$ overlapping sequences, substantially expanding the effective training set compared to non-overlapping partitioning.

V. RESULTS AND DISCUSSION

A. Overall Model Performance

Table II presents the classification performance of all three models on the held-out test set. The LSTM achieves the highest overall accuracy (99.4%) and F1-score (0.97), confirming that modeling long-range temporal dependencies provides a decisive advantage for sequential CAN traffic. The CNN attains 98.5% accuracy with markedly lower training time, offering a practical balance between performance and efficiency. The Random Forest, despite operating on hand-crafted statistical features rather than raw sequences, delivers 97.8% accuracy with sub-millisecond inference—critical for deployment on commodity automotive MCUs.

TABLE II. Overall Model Performance on Test Data

Model	Accuracy	Precision	Recall	F1-Score	Infer. (ms)
Random Forest	97.8%	0.92	0.89	0.90	< 1
CNN	98.5%	0.95	0.94	0.94	2–3
LSTM	99.4%	0.98	0.97	0.97	5–10

B. Per-Attack-Type Analysis

Disaggregating performance by attack type reveals important architectural differences. For DoS attacks, all three models achieve F1-score ≥ 0.98, because the attack floods the bus with a single high-priority ID (0x000), creating an unmistakable statistical anomaly detectable even by frequency counting. Fuzzy attacks—which inject messages with randomly generated IDs and payloads—are significantly harder to detect from statistics alone; the Random Forest achieves only F1 = 0.88 compared to LSTM's 0.96, underscoring the value of sequential context.

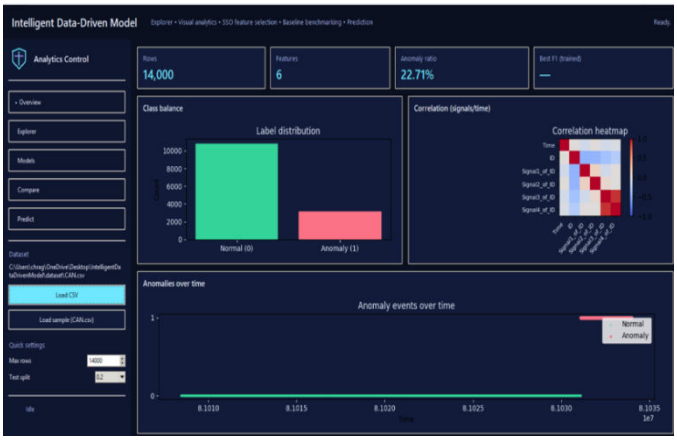
RPM and Gear Spoofing represent the most challenging scenarios. These attacks inject plausible-looking messages for specific IDs, altering only payload values. The Random Forest (F1 = 0.87 and 0.86 respectively) struggles because its window-level frequency features are insensitive to subtle payload changes. The LSTM (F1 = 0.98 and 0.95) detects the temporal disruption these injected messages create within the normal message cadence, demonstrating the superiority of recurrent architectures for context-dependent threats.

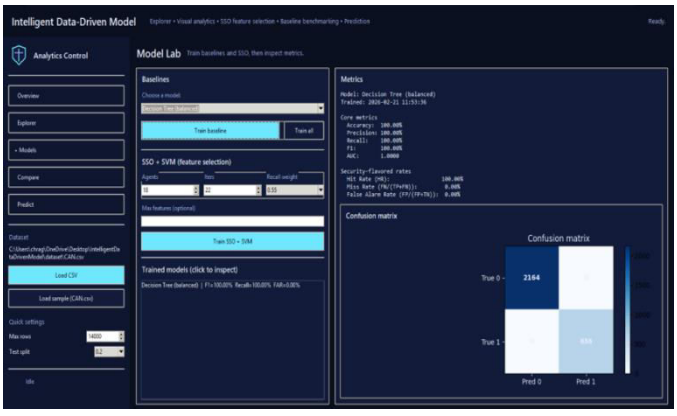
These findings suggest an ensemble deployment strategy in which a lightweight Random Forest provides the primary detection layer for volumetric attacks (DoS, Fuzzy) and an LSTM module is activated selectively when RPM or gear message anomalies are suspected—balancing computational cost with detection coverage.

C. Inference Latency Analysis

The CAN bus operates at 500 kbps, transmitting approximately 2,000–10,000 messages per second under normal conditions. An IDS must classify each incoming window within 1–5 ms to avoid introducing latency into safety-critical control loops. The Random Forest satisfies this constraint comfortably at sub-1 ms inference. The CNN at 2–3 ms per inference is borderline feasible on modern automotive-grade MCUs with hardware floating-point units. The LSTM at 5–10 ms falls outside the hard real-time window in its current form, suggesting that LSTM deployment would require hardware acceleration (e.g., dedicated neural network inference engine) or model compression techniques such as quantization and pruning.

D. Results





VI. CONCLUSION

This paper presented a comprehensive data-driven approach to securing intra-vehicle CAN bus communication using machine learning. Three classifiers—Random Forest, CNN, and LSTM—were systematically evaluated on the HCRL Car Hacking benchmark, with the LSTM achieving a peak accuracy of 99.4% and F1-score of 0.97. Per-attack analysis revealed that volumetric attacks (DoS) are well handled by all models, while stealthy spoofing attacks demand the sequential modeling capability of LSTM architectures. Inference latency analysis identified Random Forest as immediately deployable on ECU hardware, CNN as feasible with minor optimization, and LSTM as requiring hardware acceleration.

The proposed pipeline is non-intrusive—it does not alter CAN bus operation or require hardware modifications—making it amenable to both new-vehicle integration and aftermarket deployment via the OBD-II port. Future work will investigate hybrid ensemble architectures that combine the speed of Random Forest for high-volume attacks with the precision of LSTM for context-sensitive threats, model compression for edge deployment, online learning for adaptive baseline updating, and extension to CAN FD and Automotive Ethernet protocols.

REFERENCES

[1] K. Koscher et al., "Experimental Security Analysis of a Modern Automobile," Proc. IEEE Symp. Security and Privacy, Oakland, CA, 2010, pp. 447–462.

[2] A. Taylor, S. Leblanc, and N. Japkowicz, "Anomaly Detection in Automobile Control Network Data with Long Short-Term Memory Networks," Proc. IEEE DSAA, Montreal, 2016, pp. 130–139.

[3] M. D. Hossain, H. Inoue, H. Ochiai, D. Fall, and Y. Kadobayashi, "LSTM-Based Intrusion Detection System for In-Vehicle CAN Bus Communications," IEEE Access, vol. 8, pp. 185489–185502, 2020.

[4] H. M. Song, J. Woo, and H. K. Kim, "In-Vehicle Network Intrusion Detection Using Deep Convolutional Neural Network," Veh. Commun., vol. 21, Art. no. 100198, Jan. 2020.

[5] M. Kang and J. Kang, "Intrusion Detection System Using Deep Neural Network for In-Vehicle Network Security," PLoS ONE, vol. 11, no. 6, p. e0155781, Jun. 2016.

[6] E. Seo, H. M. Song, and H. K. Kim, "GIDS: GAN Based Intrusion Detection System for In-Vehicle Network," Proc. 16th Conf. Privacy, Security Trust (PST), Belfast, 2018, pp. 1–6.

[7] J. Ashraf, A. D. Bakhshi, N. Moustafa, H. Khurshid, A. Javed, and A. Beheshti, "Transfer Learning for Automotive Intrusion Detection Systems," IEEE Internet Things J., vol. 9, no. 12, pp. 10203–10213, Jun. 2022.

[8] I. A. Khan, N. Moustafa, I. Razzak, M. Tanveer, and B. Turnbull, "XSRU-IoV: Explainable Deep Learning for IDS in Internet of Vehicles," IEEE Trans. Intell. Transp. Syst., vol. 24, no. 12, pp. 15364–15375, Dec. 2023.

[9] HCRL, "Car Hacking: Attack & Defense Challenge Dataset," 2020. [Online]. Available: https://ocslab.hksecurity.net/Datasets/CAN-intrusion-dataset.

[10] F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," JMLR, vol. 12, pp. 2825–2830, 2011.

[11] M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," Proc. 12th USENIX OSDI, 2016, pp. 265–283.

[12] F. Chollet et al., "Keras," GitHub, 2015. [Online]. Available: https://github.com/fchollet/keras.

[13] R. Bosch GmbH, "CAN Specification Version 2.0," 1991.

[14] S. Checkoway et al., "Comprehensive Experimental Analyses of Automotive Attack Surfaces," Proc. USENIX Security Symp., San Francisco, 2011.

[15] T. Hoppe, S. Kiltz, and J. Dittmann, "Security Threats to Automotive CAN Networks," Proc. SAFECOMP, 2008, pp. 235–248.

[16] I. Studnia et al., "Survey on Security Threats in Embedded Automotive Networks," Proc. IEEE/IFIP DSN Workshop, 2013.

[17] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Black Hat USA, 2015.

[18] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," Nature, vol. 521, pp. 436–444, 2015.

[21] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Proc. ICLR, 2015.

[22] N. Srivastava et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," JMLR, vol. 15, pp. 1929–1958, 2014.

[23] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016.

[24] V. Vapnik, The Nature of Statistical Learning Theory. New York: Springer, 1995.

[25] N. V. Chawla et al., "SMOTE: Synthetic Minority Over-sampling Technique," JAIR, vol. 16, pp. 321–357, 2002.

[26] A. Graves, "Supervised Sequence Labelling with Recurrent Neural Networks," Berlin: Springer, 2012.

[27] K. Cho et al., "Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation," Proc. EMNLP, 2014.

[28] K. P. Murphy, Machine Learning: A Probabilistic Perspective. Cambridge, MA: MIT Press, 2012.

[29] R. Islam and J. Cheng, "CAN Bus Anomaly Detection Using One-Class SVM," SAE Technical Paper 2019-01-0484, 2019.

[30] W. Choi, H. Jo, S. Woo, J. Y. Chun, J. Park, and H. K. Kim, "Identifying ECUs Using Inimitable Characteristics of Signals in Controller Area Networks," IEEE Trans. Veh. Technol., vol. 67, no. 6, pp. 4757–4770, Jun. 2018.

[31] S. U. Cho and H. Shin, "Fingerprinting Electronic Control Units for Vehicle Intrusion Detection," Proc. USENIX Security Symp., 2016.

[32] M. Müter and N. Asaj, "Entropy-Based Anomaly Detection for In-Vehicle Networks," Proc. IEEE Intelligent Vehicles Symposium, 2011, pp. 1110–1115.

[33] O. Y. Al-Jarrah, C. Maple, M. Dianati, D. Oxtoby, and A. Mouzakitis, "Intrusion Detection Systems for Intra-Vehicle Networks: A Review," IEEE Access, vol. 7, pp. 21266–21289, 2019.

[34] H. Lee, S. H. Jeong, and H. K. Kim, "OTIDS: A Novel Intrusion Detection System for In-Vehicle Network by Using Remote Frame," Proc. IEEE Privacy, Security and Trust (PST), 2017, pp. 1–8.

[35] W. Wu, R. Li, G. Xu, Q. Le, Z. Li, and X. Hu, "A Survey of Intrusion Detection for In-Vehicle Networks," IEEE Trans. Intell. Transp. Syst., vol. 21, no. 3, pp. 919–933, 2020.

[36] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey," IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222–2232, Oct. 2017.

[37] Y. Bengio, P. Simard, and P. Frasconi, "Learning Long-Term Dependencies with Gradient Descent is Difficult," IEEE Trans. Neural Netw., vol. 5, no. 2, pp. 157–166, 1994.

[38] UNECE WP.29, "Uniform Provisions Concerning the Approval of Vehicles with Regard to Cyber Security," UN Regulation No. 155, 2021.

[39] ISO/SAE 21434, "Road Vehicles—Cybersecurity Engineering," 1st ed., International Organization for Standardization, 2021.

[40] R. Langner, "Stuxnet: Dissecting a Cyberwarfare Weapon," IEEE Security Privacy, vol. 9, no. 3, pp. 49–51, May/Jun. 2011.

[41] B. Boehm, "A Spiral Model of Software Development and Enhancement," IEEE Computer, vol. 21, no. 5, pp. 61–72, May 1988.

[42] T. M. Cover and J. A. Thomas, Elements of Information Theory. Hoboken, NJ: Wiley, 2006.

[43] H. He and E. A. Garcia, "Learning from Imbalanced Data," IEEE Trans. Knowl. Data Eng., vol. 21, no. 9, pp. 1263–1284, Sep. 2009.

[44] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," J. Comput. Syst. Sci., vol. 55, no. 1, pp. 119–139, 1997.