vehicles

MDPI

Article

Temporal-Window-Aware Physics-Informed Edge IDS for Multi-Class IoV Misbehavior Detection Under Ideal and Realistic BSM Observability

Abdelhabib Bourouis [1,*], Ahlem Nasri [1], Sofiane Zaidi [1], Liamine Bekhouche [1] and Carlos T. Calafate [2]

[1] Research Laboratory on Computer Science's Complex Systems ReLa(CS)^2, Department of Mathematics and Computer Science, University of Oum El Bouaghi, Oum El Bouaghi 04000, Algeria; nasri.ahlem@univ-oeb.dz (A.N.); zaidi.sofiane@univ-oeb.dz (S.Z.); bekhouche.liamine@univ-oeb.dz (L.B.)

[2] Department of Computer Engineering (DISCA), Universitat Politècnica de València, 46022 Valencia, Spain; calafate@disca.upv.es

* Correspondence: bourouis.abdelhabib@univ-oeb.dz

Abstract

The Internet of Vehicles (IoV) relies on Basic Safety Messages (BSMs) for cooperative awareness, yet these broadcasts remain vulnerable to falsification, replay, flooding, Sybil-based, and motion-manipulation attacks. This paper proposes a temporal-window-aware physics-informed edge-oriented Intrusion Detection System (IDS) for 20-class IoV misbehavior detection under two simulation-based BSM observability regimes: ideal noise-free kinematics and realistic noise-inclusive observables reconstructed using the sensor-error components supplied separately by VeReMi Extension. Accordingly, "realistic" denotes a noise-inclusive simulation condition rather than real-world validation. From VeReMi Extension streams, the framework derives a compact 20-feature representation capturing kinematics, timing, replay cues, pseudonym dynamics, position-consistency residuals, zero-pattern behavior, and long-horizon motion indicators. These features are normalized with a training-only robust scaler, organized into sender-specific temporal windows, and classified using a lightweight three-layer stacked Long Short-Term Memory (LSTM) with residual temporal pooling. Four implementation variants are evaluated: dense Keras, default-optimized TensorFlow Lite, pruning-only Keras, and pruning-plus-compression TensorFlow Lite. Temporal sensitivity identifies $T = 40$ as the best robustness–latency compromise under the realistic noise-inclusive regime. At $T = 40$, the final pruned-and-compressed TensorFlow Lite model achieves 99.60% accuracy and 99.09% macro-F1 under ideal observability, and 99.38% accuracy and 98.67% macro-F1 under realistic noise-inclusive observability, with an 88.38 KB footprint and 0.1283 ms controlled-runtime latency. Large-scale Central Processing Unit (CPU) benchmarks on 150,000 noise-inclusive test sequences provide a platform-dependent runtime reference, with the pruned TensorFlow Lite model reaching 99.14% accuracy, 98.18% macro-F1, and 3.544 ms average latency on a multi-core Intel Xeon CPU. To complement this high-throughput evaluation, edge-deployment potential is profiled using the official C++ TensorFlow Lite benchmark tool. When evaluated using a single CPU thread without batching, the final artifact achieves an unbatched per-sequence latency of 1.356 ms, corresponding to less than 1.4% of the standard 100 ms BSM generation interval. An architecture-width ablation identifies the 64/32/32 recurrent stack as the performance–resource knee point: expanding it to 128/64/64 improves validation macro-F1 by only 0.0019 percentage points while increasing TensorFlow Lite footprint and latency by factors of 2.46 and 2.32, respectively. A training-time architecture-preserving feature-family ablation confirms that engineered descriptors are essential: raw kinematics

Check for updates

Academic Editor: Yang Song

Received: 21 July 2026
Revised: 1 September 2026
Accepted: 4 September 2026
Published: 9 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

alone reduce noise-inclusive macro-F1 from 98.67% to 67.49%, with pseudonym dynamics and position-consistency cues producing the largest individual degradations.

Keywords: Internet of Vehicles; VANET Security; Basic Safety Messages; VeReMi Extension; Intrusion Detection System; Edge Computing; LSTM; Model Pruning; TensorFlow Lite

1. Introduction

The rapid evolution of Intelligent Transportation Systems (ITS) is accelerating the transition from conventional Vehicular Ad-hoc Networks (VANETs) toward the Internet of Vehicles (IoV), where vehicles, roadside infrastructure, and external digital services interact over a dynamic and open wireless ecosystem [1]. In this environment, vehicles continuously exchange Basic Safety Messages (BSMs) containing position, speed, acceleration, and heading information to support cooperative awareness and safety-critical services such as collision avoidance, lane-change assistance, emergency braking, and traffic-flow coordination [2]. However, this openness also increases the attack surface of vehicular communications. Recent surveys show that IoV systems remain vulnerable to falsification, spoofing, replay, flooding, Sybil-based impersonation, and denial-of-service behaviors, all of which can compromise safety and trust in connected transportation [3–6].

Intrusion Detection Systems (IDSs) have therefore become an essential complementary defense layer for vehicular networks. Machine Learning (ML) and Deep Learning (DL) models can capture complex malicious patterns that are difficult to describe with static rules alone [6]. Among them, Long Short-Term Memory (LSTM) networks are particularly relevant because BSM streams are sequential: the legitimacy of a message depends not only on its instantaneous values, but also on its consistency with the sender's recent trajectory. This temporal aspect is important under standardized V2X conditions, where BSMs are generated at approximately 10 Hz according to SAE J2945/1 and follow the message structure formalized in SAE J2735 [7,8]. As in lightweight in-vehicle architectures [9], an edge-oriented vehicular IDS must therefore combine temporal sequence modeling with very low execution latency.

Despite recent progress, several limitations remain. First, many vehicular IDSs still focus on binary anomaly detection, although different attack families require different mitigation strategies [6]. Distinguishing replay, falsification, disruptive manipulation, flooding, and Sybil-based behaviors under a unified sequential model is more operationally useful than a generic malicious/non-malicious alarm [10,11]. Second, many multi-class IoV IDSs rely on heavy ensembles, hybrid deep architectures, self-attention, federated training, or synthetic augmentation, which may conflict with the memory, latency, and energy constraints of embedded On-Board Units (OBUs) [12–16]. This issue also appears in distributed feature-fusion approaches based on Transformer autoencoders and CNN-LSTM attention blocks [17], and in hierarchical binary decomposition strategies that can increase inference complexity at the edge [18].

This deployment gap is reinforced by the cost of modern neural models. Model-compression surveys emphasize that high DL accuracy is often accompanied by prohibitive storage and computation requirements [19,20]. More sophisticated pruning–quantization–distillation pipelines and teacher–student compression strategies have been proposed [21–23], but their multi-stage optimization may be unnecessarily complex for a deployment-driven vehicular IDS. At the same time, recent IoV studies still report strong performance with models that remain large, insufficiently compressed, or dependent on external infrastructure, federated coordination, or cloud-assisted inference [24–27]. This

leaves a gap between algorithmic performance and practical deployability on embedded vehicular edge platforms.

A further challenge concerns dataset realism, class imbalance, and BSM observability. Many automotive IDS studies rely on generic IT or in-vehicle datasets that do not preserve the physical semantics of inter-vehicular BSM exchange [3]. VeReMi Extension provides a BSM-centric simulation benchmark with 20 behavioral classes, including replay, disruptive, DoS, and Sybil-related attacks [28,29]. However, its strong class imbalance has often encouraged Synthetic Minority Over-sampling Technique (SMOTE), Generative Adversarial Network (GAN)-based augmentation, or related synthetic balancing strategies [30–32]. Such methods may improve minority-class statistics but can also generate kinematically implausible samples. Recent VeReMi-oriented studies instead suggest that temporal discontinuities, kinematic plausibility, and traffic-level behavioral effects provide more physically grounded cues for vehicular anomaly detection [33–35]. Beyond imbalance, another evaluation issue is whether the detector receives clean simulator-side kinematic fields or noise-inclusive BSM-derived observables. The former provides an optimistic upper-bound reference, whereas the latter better imitates receiver-side measurement uncertainty within the simulation; neither constitutes real-world vehicle or over-the-air V2X validation.

Motivated by these limitations, this paper proposes a temporal-window-aware physics-informed edge-oriented IDS for 20-class IoV misbehavior detection. The system is evaluated under two complementary observability regimes derived from the same VeReMi Extension simulation. The *ideal* regime uses clean simulator-side kinematic fields as an upper-bound reference. The *realistic* regime is a simulation-based, noise-inclusive condition constructed from the corresponding clean fields and dataset-provided sensor-error components. It approximates receiver-side measurement uncertainty more closely than the ideal regime, but it is not equivalent to validation with physical vehicles, OBUs, RSUs, or operational wireless links.

Instead of relying on synthetic balancing, the framework derives 20 sender-local descriptors from VeReMi Extension streams. These descriptors capture kinematics, timing, replay cues, pseudonym changes, position-consistency residuals, zero-pattern behavior, speed variation, and long-horizon constant-motion indicators. The resulting features are normalized with a training-only robust scaler. They are then organized into sender-local temporal windows with $T \in \{10, 20, 30, 40, 50\}$. Classification is performed using a lightweight three-layer stacked LSTM with residual temporal pooling.

The evaluation uses sender-level splitting to reduce trajectory leakage. It compares four implementation variants: Dense Keras, Compression-only TensorFlow Lite, Pruning-only stripped Keras, and Pruning + Compression TensorFlow Lite. Rather than optimizing accuracy alone, the study jointly analyzes robustness to noise-inclusive observability, temporal-context length, class-balanced performance, serialized footprint, and inference latency. This design assesses whether a compact physics-informed recurrent IDS can preserve fine-grained 20-class diagnosis under the simulation-based realistic BSM regime while providing a measurable edge-deployment potential.

Compared with recent VeReMi-oriented IDSs, the proposed work addresses a different combination of requirements. Youness et al. [26] report strong 20-class performance with engineered features and Bi-LSTM modeling, but rely on SMOTE-based balancing and do not provide a kilobyte-level TensorFlow Lite deployment artifact. Cao et al. [25] explore an edge–cloud split-learning architecture with a vehicle-side component, but the setting is binary anomaly detection and remains dependent on a cloud-side LLM. Federated and Transformer-based approaches further improve distributed or attack-wise detection, but typically do not combine unified 20-class diagnosis, sender-level leakage prevention,

explicit ideal-versus-realistic BSM observability, temporal-window sensitivity, and compressed standalone edge inference in a single pipeline. The present work is designed to close this specific gap.

The main contributions of this paper go beyond aggregate classification performance. They address six coupled requirements: simulation-based noise-inclusive observability, fine-grained diagnosis, physics-informed representation, leakage-aware temporal evaluation, temporal and architectural trade-off analysis, and edge-oriented model export:

- **Dual-observability BSM evaluation:** the same 20-class IDS pipeline is evaluated under ideal noise-free observability and simulation-based realistic (noise-inclusive) observability. This controlled comparison quantifies the effect of incorporating the dataset-provided sensor-error components while explicitly distinguishing the noise-inclusive benchmark condition from real-world vehicle or wireless-link validation.
- **Physics-informed representation without synthetic balancing:** a compact 20-feature sender-local descriptor set captures kinematics, timing, replay cues, pseudonym changes, position consistency, zero patterns, speed changes, and long-horizon constant-motion behavior while preserving the original class distribution. This avoids SMOTE/GAN-style trajectory fabrication while keeping the learned decision boundaries tied to physically plausible BSM histories.
- **Leakage-aware multi-class diagnosis:** the proposed system performs unified 20-class VeReMi Extension classification under sender-level partitioning, thereby reducing trajectory leakage and enabling the assessment of generalization to unseen vehicles.
- **Standalone edge-oriented TensorFlow Lite artifact:** the export path combines 30% magnitude pruning with TensorFlow Lite conversion to produce a compact standalone artifact. The resulting model does not rely on cloud assistance, split-learning inference, or federated coordination at prediction time. Unlike more sophisticated adaptive-pruning, quantization-aware, and distillation pipelines [21–23,36,37], the proposed path prioritizes reproducibility, artifact portability, and measurable footprint–latency behavior. In the simulation-based realistic regime, the final artifact has a serialized footprint of 88.38 KB and achieves an average latency of 3.544 ms in the large-scale Python CPU benchmark and a 1.356 ms in the official single-thread C++ TensorFlow Lite benchmark tool. These measurements support its edge-deployment potential under the evaluated runtime conditions; validation on representative OBU or roadside-edge hardware remains necessary before making production-level deployment claims.
- **Temporal and resource trade-off analysis:** the study evaluates $T \in \{10, 20, 30, 40, 50\}$ to quantify the impact of temporal context length. It also compares dense training, TensorFlow Lite conversion, pruning-only optimization, and pruning-plus-compression using accuracy, macro-F1, weighted-F1, serialized size, and inference latency. The retained configuration is $T = 40$, which provides the best robustness–latency compromise in the simulation-based realistic regime while preserving near-peak performance under ideal observability. A recurrent-width ablation additionally compares 32/16/16, 64/32/32, and 128/64/64 configurations and identifies 64/32/32 as the performance–resource knee point under the evaluated protocol.
- **Feature-family ablation of the final exported artifact:** an architecture-preserving training-time ablation is conducted using the retained simulation-based realistic $T = 40$ configuration. The contribution of each descriptor family is quantified under the same model architecture, window length, pruning target, and TensorFlow Lite conversion pipeline. Raw kinematics alone reduce macro-F1 from 98.67% to 67.49%, while pseudonym dynamics and position-consistency cues produce the largest individual degradations.

Figure 1 provides a graphical overview of the proposed IDS pipeline, from VeReMi Extension BSM streams and physics-informed feature extraction to temporal-window-aware LSTM classification, deployment-oriented compression, and evaluation under the two simulation-based observability regimes (ideal and realistic).

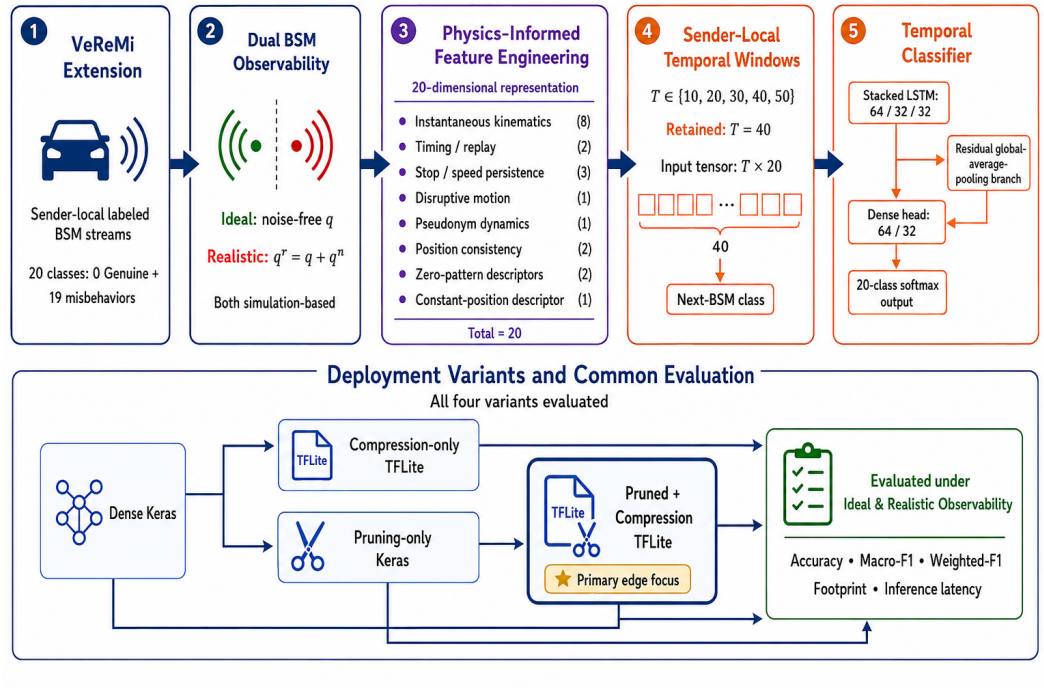


Figure 1. Overview of the proposed temporal-window-aware physics-informed IDS. Each sender-local BSM stream is converted into a 20-dimensional representation comprising instantaneous kinematics, timing/replay, stop/speed-persistence, disruptive-motion, pseudonym-dynamics, position-consistency, zero-pattern, and constant-position descriptors. Temporal windows $T \in \{10, 20, 30, 40, 50\}$ are evaluated, and $T = 40$ is retained as the deployment-oriented configuration. Each $T \times 20$ sequence is processed by a 64/32/32 stacked LSTM with a residual global-average-pooling branch, followed by a 64/32 dense classification head and a 20-class softmax output. The four evaluated deployment variants are Dense Keras, Compression-only TensorFlow Lite, Pruning-only Keras, and Pruning + Compression TensorFlow Lite.

Together, these contributions position the proposed IDS as a compact, observability-aware, and edge-oriented solution with a measurable deployment potential for fine-grained BSM misbehavior detection, rather than as another accuracy-only VeReMi Extension classifier.

The remainder of this paper is organized as follows. Section 2 reviews vehicular IDS datasets, evaluation realism, architectural families, class-imbalance mitigation, synthetic augmentation limits, and edge-oriented deployment paradigms. Section 3 presents the proposed methodology, including VeReMi Extension, dual BSM observability, sender-level leakage prevention, physics-informed feature engineering, temporal window construction, LSTM modeling, compression variants, and evaluation metrics. Section 4 reports sequence construction, temporal-window and threshold sensitivity, aggregate performance, deployment trade-offs, architecture-width and temporal-model comparisons, sparsity sensitivity, runtime benchmarks, individual and combined feature-group ablations, and class-wise behavior. Section 5 discusses observability, deployment trade-offs, feature design, comparison with existing IDSs, and threats to validity. Finally, Section 6 concludes the paper and outlines future work.

2. Related Work

Recent research on vehicular Intrusion Detection Systems (IDSs) has progressed rapidly with the rise of the Internet of Vehicles (IoV) and Cooperative Intelligent Transport Systems (C-ITS). However, the literature remains fragmented across four main directions: dataset realism and BSM observability, architectural choices for temporal intrusion detection, class-imbalance mitigation, and edge-oriented deployment. Many studies still rely on generic network or intra-vehicle benchmarks, optimize binary detection rather than fine-grained diagnosis, or propose federated and hybrid models without fully reporting embedded deployment constraints. In addition, two aspects remain underexplored: whether BSM-based IDSs are evaluated with clean simulator-side kinematic fields or noisy receiver-side observables, and how temporal window length affects the trade-off between detection performance and inference cost. The following subsections position the present work along these dimensions.

2.1. Vehicular IDS Datasets and Evaluation Realism

A first distinction concerns the realism of the evaluation benchmark. Although IoV security is governed by vehicular spatio-temporal semantics, many IDS studies still use generic network corpora, intra-vehicle Controller Area Network (CAN) traces, or mixed traffic abstractions that do not preserve inter-vehicular Basic Safety Message (BSM) behavior. Recent surveys highlight this mismatch: while automotive IDS research often focuses on CAN or Ethernet, realistic V2X intrusion detection must also account for mobility, temporal continuity, communication periodicity, and motion plausibility [3–6]. This limitation is visible in IoV-oriented frameworks evaluated on CAN-intrusion, CICIDS2017, CSE-CIC-IDS-2018, or CIC-DDoS-2019 datasets [38,39]. Such benchmarks are useful for studying CAN message injection or external network attacks, but they do not capture trajectory continuity, pseudonym dynamics, and cooperative-awareness semantics in BSM exchange.

In contrast, the VeReMi family of benchmarks provides a more appropriate foundation for inter-vehicular misbehavior detection because it preserves BSM semantics rather than reducing vehicular security to generic IP traffic [2]. This line has been extended by VeReMiAP, which integrates CAM-based Fake Reporting Attacks and models downstream hazard propagation [33]. The original VeReMi dataset introduced a public basis for reproducible VANET misbehavior detection, while VeReMi Extension added a realistic sensor-error model, a broader attack library, and larger observation volumes under the LuST traffic scenario [28]. This makes VeReMi Extension particularly suitable for studying replay, disruptive, flooding, and Sybil-related behaviors under vehicular motion and cooperative-awareness constraints, in line with recent IoV/vehicular IDS surveys emphasizing the need for realistic V2X-oriented datasets [5,6].

Dataset realism also depends on the *evaluation protocol*. In sequential vehicular data, random splitting can leak temporally adjacent BSMs from the same sender across training and testing partitions, allowing the model to learn sender- or trajectory-specific regularities rather than generalizable attack signatures. A realistic vehicular IDS should therefore be evaluated on *unseen trajectories and unseen senders*, not merely on shuffled messages. This concern is consistent with recent VANET IDS surveys, which stress that reported detection scores should be interpreted together with validation realism, simulation assumptions, time complexity, scalability, and energy-efficiency constraints [40].

A final evaluation dimension concerns BSM observability. Simulation-based datasets often provide clean ground-truth kinematic fields, which are useful for estimating an upper-bound diagnostic performance. A receiver-side IDS, however, would observe variables affected by measurement noise, localization uncertainty, and communication impairments. The present work therefore distinguishes *ideal BSM observability*, based on clean simulator-

side fields, from *simulation-based realistic BSM observability*, based on noise-inclusive fields reconstructed using the sensor-error components supplied by VeReMi Extension. The latter represents a controlled approximation of receiver-side measurement uncertainty within the simulation, not real-vehicle or over-the-air validation. The dual-observability protocol consequently evaluates the same IDS as both an upper-bound reference and a more deployment-oriented benchmark configuration.

2.2. Architectural Families for Vehicular Intrusion Detection

A second axis concerns the *type of IDS architecture* used to model vehicular attacks. The literature can be grouped into four main families: *rule-based plausibility models*, *classical machine learning and ensemble approaches*, *deep sequential architectures*, and *attention-based or hybrid large-scale models*.

The first family relies on *local plausibility checks or filtering-based constraints* over motion variables. For example, LA-DETECTS uses Kalman and Hampel filtering to detect implausible vehicular behavior directly from message content [41]. These methods are lightweight, interpretable, and often deployable with low computational overhead. However, fixed or quasi-fixed behavioral assumptions may limit their ability to detect stealthy, progressive, or replay-like attacks whose maliciousness emerges only over a sequence rather than from a single BSM.

The second family includes *classical ML and ensemble-based methods*. Two earlier studies are particularly relevant because they document and use the same labeled tabular derivative employed in the present work. Slama et al. [42] compared conventional machine- and deep-learning classifiers on VeReMi Extension behaviors using attack-wise evaluations and a random 70/30 message-level split. Subsequently, Slama et al. [43] investigated binary, three-class, original 20-class, and guided nine-group formulations on the prepared 3,194,808-record dataset. In their original 20-class formulation, Random Forest with recursive feature elimination reported precision, recall, and F1-score values of 71.0%, 69.1%, and 69.6%, respectively. These studies establish the preparation and early classification lineage of the labeled dataset, but their message-level, attack-wise, or regrouped-class protocols differ from the sender-disjoint, unified 20-class temporal evaluation adopted here. Slama et al. [18] proposed a One-Versus-All Binary Tree strategy that decomposes the 20-class VeReMi problem into hierarchical binary decisions. Although this can improve separability for some classes, it may also increase inference complexity and error propagation. Ensemble methods such as CatBoost [12], multi-stage random-forest-based frameworks [44], and explainable RF–CatBoost–LR stacking with SHAP [45] can provide strong tabular performance, but they generally offer limited temporal modeling and often lack concrete evidence of compressed, low-latency deployment on memory-constrained vehicular hardware.

The third family, which is the most relevant for BSM-centric intrusion detection, consists of *deep sequential architectures*. Recent surveys emphasize the growing role of recurrent and temporal DL models in vehicular IDS [6,40]. LSTM-based models are particularly suitable because successive BSMs are temporally correlated, and attacks such as replay, eventual stop, disruptive falsification, or progressive motion manipulation may only become visible through sender-local history. Kapoor and Prasad [30] combined CNN and LSTM blocks on VeReMi Extension; Zeng and Mi [11] proposed a robust LSTM within a federated setting, while Youness et al. [26] showed that engineered vehicular features combined with Bi-LSTM layers can achieve strong multi-class results. Mahesh et al. [46] recently proposed a HyperLSTM-based anomaly detector on VeReMi Extension, using overlapping temporal windows and dropout regularization to improve sequential anomaly detection, and reported 98% accuracy for binary normal/attack classification. However,

the evaluation remains binary, and single-observability does not address unified 20-class VeReMi Extension diagnosis, nor does it report a compressed TensorFlow Lite artifact, serialized footprint, or OBU-oriented inference benchmark. Alladi et al. [47] also provide an important early deep-learning baseline on VeReMi Extension for C-ITS misbehavior classification. They propose single-stage and two-stage Deep Learning Classification Engines based on LSTM and CNN–LSTM models, and evaluate 18 behavior types using sequences constructed from ground-truth files. Their work is relevant because it explicitly connects VeReMi Extension classification with edge-server execution on Jetson Nano and Raspberry Pi testbeds. However, it differs from the present study in several respects: it considers 18 rather than 20 behavior classes, uses overlapping 20-step sequences with a slide length of 10, does not separate ideal simulator-side observability from noisy receiver-side BSM observability, and does not report a compressed TensorFlow Lite artifact with sender-level leakage prevention and feature-family ablation. Meegammana and Fernando [48] further evaluated SimpleRNN, LSTM, and GRU models with attention on a balanced VeReMi Extension subset for binary DoS-oriented V2X detection, reporting accuracy, F1-score, latency, CPU usage, memory usage, and Keras/TFLite footprint. ADVENT also exploits time slicing within federated recurrent structures for DDoS-oriented VeReMi Extension detection [49]. These works confirm the relevance of recurrent modeling, but many remain limited to binary or attack-specific settings, balanced subsets, federated evaluation, or incomplete embedded-compression evidence. Hoang et al. [10] also explored a semi-supervised convolutional adversarial autoencoder for in-vehicle CAN intrusion detection, reporting reduced trainable parameters and inference time, but the setting remains binary and intra-vehicle rather than BSM-centric multi-class IoV misbehavior detection.

The fourth family includes *attention-based and hybrid large-scale models*. Transformer-inspired IDSs such as the 5G edge misbehavior detection framework of Missara et al. [13] exploit self-attention and report strong results on VeReMi Extension attack datasets, but they evaluate attack-wise subsets rather than the same unified 20-class softmax setting. Similarly, DAGSNet combines Vision Transformers, wavelet transforms, graph attention, convolutional backbones, and cryptographic mechanisms such as AES-256, secure multi-party computation, and homomorphic encryption [14]. Such architectures can be highly accurate, but they illustrate a recurring problem of architectural inflation: their model composition, feature-extraction complexity, and security machinery are difficult to reconcile with kilobyte-level memory budgets and micro-latency requirements on embedded OBUs.

Overall, the literature shows a tension between diagnostic granularity and deployment simplicity. Rule-based and binary systems are easier to execute but provide limited operational insight, while large hybrid and attention-based models often remain difficult to embed. Sequential recurrent models offer a practical middle ground, especially when combined with compact physics-informed features and explicit compression.

2.3. Class Imbalance Mitigation and the Limits of Synthetic Augmentation

A third challenge in vehicular IDS research is the *severe class imbalance* of realistic datasets. In VeReMi Extension and related benchmarks, normal traffic dominates the distribution, while several malicious behaviors remain rare but safety-critical. This imbalance has motivated many *data-level balancing* strategies, especially oversampling and synthetic sample generation.

SMOTE is among the most common solutions. Kapoor and Prasad [30] combined SMOTE with a CNN-LSTM pipeline for VeReMi Extension, while Choudhary et al. [32] used SMOTE-ENN with random forests and XGBoost to improve minority-class representation. Beyond interpolation-based oversampling, Xu et al. [31] proposed a GAN-like sample-augmentation framework for IoV zero-day attack detection. These approaches can

improve aggregate scores and stabilize training under skewed class distributions. However, imbalance is often coupled with label scarcity and noisy minority classes rather than being only a sampling problem. Nguyen et al. [50], for example, proposed a semi-supervised in-vehicle IDS based on a variational autoencoder and adversarial reinforcement learning, and highlighted that resampling may amplify noise when minority classes contain outliers. Although their study targets intra-vehicle CAN traffic, this observation remains relevant to BSM-based IDSs.

However, in the vehicular context, synthetic augmentation raises an additional concern: physical plausibility. A synthetic minority sample may be statistically useful but kinematically impossible. This is problematic for BSM-centric IDS because temporal continuity, feasible displacement, speed variation, heading evolution, and behavioral persistence are part of the attack semantics. Interpolated or generated trajectories may violate these constraints and bias the classifier toward unrealistic decision boundaries. This concern is consistent with VeReMi-based studies showing that temporal leaps, spatial differentials, and trajectory-consistency descriptors are useful for exposing replay, offset, and stealthy motion attacks [33–35].

This observation motivates a different design philosophy. Instead of compensating for imbalance by fabricating additional samples, the classifier can be supported by *physics-informed features* that expose anomalous motion patterns within the original data distribution. Such a strategy preserves trajectory realism and reduces the representational burden on the model by making timing irregularities, repeated-message behavior, motion inconsistency, pseudonym changes, and constant-motion patterns explicit before temporal learning. In the present work, this principle is implemented through a compact 20-feature representation derived from sender-specific BSM streams and applied consistently under both ideal and realistic BSM observability.

2.4. Edge Deployment, Compression, and Decentralized Learning

A fourth and increasingly important axis of the literature concerns the relationship between *IoV security intelligence* and *deployment constraints*. Here, the field can again be divided into two partially overlapping directions: *decentralized/federated learning paradigms* and *hardware-aware edge optimization*.

The decentralized direction is motivated by privacy preservation, communication locality, trust management, scalability, and response-time constraints. This motivation is consistent with recent IoV security analyses emphasizing that AI-enabled and decentralized IDS mechanisms are increasingly needed to counter multi-layer IoV threats while preserving data authenticity and confidentiality [5]. Abidi et al. [51] further identify scalability, privacy preservation, robustness, and appropriate response time as core requirements for trustworthy IoV trust-management models. Federated and decentralized IDS frameworks address part of this problem by limiting raw-data sharing across vehicles, RSUs, and edge nodes. Huang et al. [52] proposed FED-IoV, a federated IDS using transformed traffic records and a pruned MobileNet-Tiny model, while Almansour et al. [53] proposed APFed with a Lightweight Depthwise Convolutional Bottleneck Network for non-IID IoV data and resource-constrained hardware. These works support the relevance of FL for privacy-aware IoV IDS, but they are mainly evaluated on CAN, CICIDS, CSE-CIC-IDS, Car-Hacking, or CAN-Train-Test datasets rather than on BSM-centric VeReMi Extension streams.

More VeReMi-oriented decentralized IDSs have also been explored. Gurjar et al. [15] evaluated RNN, LSTM, BiLSTM, RNN-LSTM, and RNN-BiLSTM models in a Flower-based federated-learning framework on VeReMi Extension, comparing FedAvg, FedProx, and FedYogi. Zeng and Mi [11] combined a robust LSTM with FedAware aggregation for non-IID client data, while Baharlouei et al. [49] studied ADVENT for distributed DoS-

oriented robustness, attack-onset detection, and malicious-node detection on VeReMi Extension. More recent work has moved toward federated feature fusion or privacy-preserving contrastive learning [16,17]. Campos et al. [54] further proposed an unsupervised federated misbehavior detector combining variational autoencoders and Gaussian mixture models while modeling vehicles as clients under realistic non-IID VeReMi-derived partitions. These studies highlight the importance of client heterogeneity and decentralized evaluation, but they generally do not provide the same combination of unified 20-class BSM diagnosis, dual-observability evaluation, TensorFlow Lite conversion, and embedded latency evidence.

Despite their conceptual appeal, decentralized approaches do not automatically provide evidence of edge deployability. Federated learning shifts part of the burden from centralized training to local computation and repeated communication rounds, raising synchronization, bandwidth, latency, and local-complexity issues in moving vehicular settings. Hybrid distributed security frameworks such as IoV-BCFL combine federated learning with blockchain-backed model recording, IPFS storage, RSA-protected intrusion logs, and smart contracts to reduce single-point-of-failure risks and strengthen forensic traceability [55]. Although IoV-BCFL reports strong results on UNSW_NB15 and Car-Hacking with reduced communication overhead, it is not evaluated on BSM-centric VeReMi Extension streams and does not provide a TensorFlow Lite or OBU-level latency profile for fine-grained 20-class misbehavior diagnosis. Thus, privacy preservation, decentralized coordination, secure logging, forensic traceability, and embedded deployability remain related but distinct objectives.

Hierarchical, distributed, and split-learning strategies represent another attempt to reduce centralized IoV intrusion-detection costs. Cao et al. [25] rely on an edge–cloud split-learning configuration, while Uddin et al. [24] proposed a three-level hierarchical IDS in which a root classifier performs benign/attack detection, second-level classifiers distinguish broad attack families such as DoS and spoofing, and third-level classifiers identify more specific spoofing subtypes. Their framework combines Boruta-based feature selection with classical ML classifiers and distributes detection across vehicles, RSUs, near-edge nodes, and the cloud. These designs are valuable from a systems perspective, but they are not evaluated on BSM-centric VeReMi Extension streams under dual observability, and they may introduce communication or cloud dependencies that become fragile under jamming, congestion, or latency-sensitive attack scenarios.

Recent model-compression studies further indicate that pruning and quantization should not be treated as independent post-processing steps. Li et al. [36] proposed A-JOPQ, an adaptive joint pruning–quantization framework based on mutual learning and multi-teacher knowledge distillation, showing that the complementarity between both mechanisms can reduce accuracy degradation. Similarly, Khan and Rafat [37] proposed Cluster-Quantized Knowledge Distillation (CQKD), which integrates structured filter pruning, layer-wise K-means weight clustering, and knowledge distillation into a unified compression-aware pipeline. These works support combining sparsification, low-precision representation, and distillation-aware training. However, because they are evaluated mainly on CIFAR-10/CIFAR-100 rather than vehicular BSM streams [36,37], they should be treated as compression-related evidence rather than direct vehicular IDS baselines.

More advanced compression pipelines combine pruning, quantization, and knowledge distillation to recover part of the accuracy lost after aggressive model reduction. PQK jointly exploits iterative pruning, quantization-aware training, and knowledge distillation by reusing weights discarded during pruning to form a teacher network without requiring a separately pre-trained teacher [21]. QRPK targets quantization-robust sparse models by training pruned networks to remain effective across multiple bit-widths, which is relevant

for adaptive edge inference but increases training complexity [22]. Recent time-series compression work also shows that pruning, mixed-precision quantization, and knowledge distillation can reduce large temporal models by more than 99% in size while preserving most predictive performance [23]. These studies confirm the value of multi-stage compression, but also illustrate why such pipelines may be unnecessarily complex for a deployment-driven vehicular IDS when a simpler pruning-plus-TensorFlow-Lite path already reaches a kilobyte-level footprint and sub-millisecond latency. Even vehicular multi-class IDS studies often stop short of aggressive compression: VeMisNet [26] reports strong VeReMi Extension performance but remains in the tens-of-megabytes regime and leaves compression and quantization as future work, while several approaches labeled as lightweight do not provide detailed evidence on serialized model size, latency, or embedded execution feasibility [17,27].

From a comparative standpoint, the literature reveals a persistent gap: existing IDSs usually optimize only part of the problem, such as *dataset realism*, *multi-class diagnostic granularity*, *privacy-preserving decentralization*, *temporal modeling*, or *actual deployability on constrained vehicular hardware*. What remains underexplored is a single vehicular IDS pipeline that jointly provides: (i) evaluation on a realistic BSM-centric benchmark, (ii) fine-grained multi-class diagnosis, (iii) physically meaningful representation without synthetic-data dependence, (iv) explicit ideal-versus-realistic BSM observability comparison, (v) temporal-window analysis as a design parameter, and (vi) compression to an embedded-compatible footprint. This is the gap targeted by the present work.

Table 1 summarizes representative VANET/IoV IDS studies by comparing their dataset realism, methodology, diagnostic granularity, observability assumptions, and reported edge-deployment evidence. The observability column indicates whether clean simulator-side fields are distinguished from noisy receiver-side or BSM-derived observables, while the "Edge Evidence" column reports concrete deployment indicators such as footprint, latency, compression, pruning, quantization, TensorFlow Lite conversion, or execution on constrained hardware. Overall, the comparison shows that existing works usually optimize only part of the problem, leaving room for a physics-informed, dual-observability, multi-class, and explicitly compressed BSM-based IDS.

Table 1. Taxonomy and critical comparison of representative recent IDS frameworks for VANET/IoV security (2022–2026). NA: Not Available; NR: Not Reported; NS: Not Separated; NT: Not Tabulated. The "Edge Evidence" column indicates whether an exportable artifact and/or runtime measurements are reported; it does not imply validation on production OBU or RSU hardware.

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Oucheikh et al. [35]	2020	Wyoming CV Pilot BSM	LSTM-AE + GWO clustering + Auto Encoder (AE)-CNN	Anomaly types	Real BSM/ not VeReMi	Partially	• Uses real BSM-like Wyoming CV Pilot data with spatio-temporal clustering, LSTM autoencoders, and AE-CNN for abrupt, intermittent, and gradual anomalies. • Not VeReMi Extension or 20-class diagnosis; described as lightweight but without serialized footprint or TFLite-style deployment evidence.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Yang et al. [38]	2022	CAN-intrusion + CI-CIDS2017	MTH-IDS hybrid known/zero-day IDS	Multi + zero-day	NR (CAN-network traffic)	Partially	• Hybrid IDS for known and zero-day attacks across CAN and vehicular network datasets, with vehicle-level sub-millisecond packet processing. • Does not evaluate VeReMi Extension, BSM trajectories, or ideal-versus-realistic BSM observability.
Hoang et al. [10]	2022	CAN Data	Semi-supervised Conv-AAE	Binary	NA (CAN)	Partially	• Semi-supervised convolutional adversarial autoencoder for in-vehicle CAN IDS with reduced trainable parameters and inference time. • Binary intra-vehicle setting, without BSM-oriented multi-class diagnosis or OBU/TFLite deployment evidence.
Slama et al. [18]	2023	VeReMi Extension	OVA Binary Tree (ML)	20	Single/NS	NR	• Performs 20-class VeReMi Extension classification through hierarchical one-versus-all binary decomposition. • May increase inference complexity and error propagation; footprint, compression, and embedded latency are not reported.
Alladi et al. [47]	2023	VeReMi Extension	DLCEs with LSTM and CNN–LSTM models	18	Single/NS	Edge-server testbeds	• Early deep-learning C-ITS misbehavior classifier using VeReMi Extension and Jetson Nano/ Raspberry Pi testbeds. • Uses 18 classes and overlapping 20-step sequences, but does not address 20-class diagnosis, dual BSM observability, leakage-aware sender splitting, or compressed TFLite deployment.
Korium et al. [39]	2024	CIC-IDS2017/ CSE-CIC-IDS2018/ CIC-DDoS2019	RF, XGBoost, CatBoost, LightGBM + Hyperopt	Multi	NA (network traffic)	Partially	• Evaluates RF, XGBoost, CatBoost, and LightGBM with feature selection, balancing, Hyperopt, and merged generic network datasets. • Reports strong accuracy/ execution-time results, but not BSM trajectories, VeReMi Extension, or dual BSM observability.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Xie et al. [55]	2024	UNSW_NB15 /Car-Hacking	Blockchain + FL + CNN	Binary	NA (network-CAN traffic)	Partially	• IoV-BCFL combines FL, blockchain-backed model recording, IPFS storage, RSA-encrypted logs, and smart contracts, with strong results on UNSW_NB15 and Car-Hacking. • Focuses on distributed detection and secure logging rather than BSM-centric VeReMi Extension, 20-class diagnosis, or TFLite-level inference.
Moushi et al. [34]	2024	VeReMi Extension	Feature re-formulation + ML ensemble	Replay-focused	Single/NS	NR	• Focuses on replay and replay-Sybil attacks using reformulated GPS/BSM features for binary and limited multi-class detection. • Does not cover the full 20-class taxonomy, dual BSM observability, serialized footprint, or embedded latency.
Huang et al. [52]	2024	Car-Hacking/ CICIDS2017 /CTAT	FED-IoV with MobileNet-Tiny + FL	Multi/ binary depending on dataset	NA (CAN-network traffic)	Partially	• FED-IoV uses transformed traffic records, FL, and pruned MobileNet-Tiny, reporting 98.51% on Car-Hacking and 97.74% on CICIDS2017 with Raspberry-Pi-oriented latency below 10 ms. • Privacy-aware and low-power, but not BSM-centric VeReMi Extension, 20-class BSM diagnosis, or dual observability.
Xu et al. [31]	2025	F2MD-generated IoV data	FLCGAN + sample augmentation	Zero-day anomaly	Single/NS	Partially	• Few-shot conditional GAN framework for zero-day IoV attack detection, evaluated on DoS, RandomSpeed, RandomPosOffset, and Disruptive attacks with 93.32% average F1-score and about 0.04 ms delay. • Does not address unified 20-class VeReMi Extension diagnosis, dual BSM observability, or compressed TensorFlow Lite deployment.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Gurjar et al. [15]	2025	VeReMi Extension	Federated recurrent models (RNN/LSTM/BiLSTM hybrids)	Multi	Single/NS	NR	• Evaluates RNN, LSTM, BiLSTM, RNN-LSTM, and RNN-BiLSTM models within a Flower-based federated-learning framework, comparing FedAvg, FedProx, and FedYogi under different client counts. • Shows that hybrid recurrent models and FedProx improve performance under heterogeneous client distributions, but does not report serialized footprint, TensorFlow Lite conversion, embedded latency, or dual-observability evaluation.
Campos et al. [54]	2025	VeReMi	Unsupervised FL with VAE + GMM	Binary/anomaly-oriented	Single/NS	NR	• Unsupervised FL misbehavior detector combining VAE, GMM, RBM initialization, and Fed+ aggregation under realistic non-IID vehicle-client partitions. • Focuses on anomaly-oriented detection rather than unified 20-class BSM diagnosis, with no TFLite or embedded-latency evidence.
Sultana et al. [41]	2025	VeReMi Extension	LA-DETECTS data-centric checks	Attack-wise	Single/NS	Local rule-based	• Local data-centric detector using Kalman innovation errors, Hampel checks, plausibility checks, and weighted scoring over VeReMi Extension attacks. • Provides attack-wise analysis rather than unified 20-class classification, with no compressed ML artifact, footprint, or TFLite deployment.
Christy et al. [44]	2025	CICIDS2017/Cloud-assisted VANET	Multi-stage RF + feature selection	Multi	NA (network traffic)	Partially	• MLIDS-RFA uses Random Forest, feature selection, and ensemble learning, reporting 96.2% accuracy and strong efficiency/scalability indicators. • Evaluated on generic network traffic rather than BSM-centric VeReMi Extension; no dual observability or TFLite deployment evidence.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Kapoor et al. [30]	2025	VeReMi Extension	CNN-LSTM + SMOTE	Binary	Single/NS	NR	• CNN-LSTM pipeline using SMOTE to address VeReMi Extension class imbalance. • Synthetic interpolation may produce kinematically implausible traces; edge footprint, compression, and latency are not quantified.
Verma et al. [12]	2025	VeReMi Extension Simple	CatBoost Ensemble	Multi	Single/NS	NR	• CatBoost-based multi-class detector on VeReMi Extension Simple with 92.3% accuracy, 0.89 macro-F1, SMOTE, LIME, and stratified validation. • Serialized footprint, embedded latency, and compression-aware deployment evidence are not reported.
Missara et al. [13]	2025	VeReMi Extension	Transformer + FL	Attack-wise	Single/NS	Edge-network architecture only	• Transformer-based 5G edge misbehavior detector with FL, reporting 99.37% accuracy, 98.69% recall, and 99.30% F1-score. • Evaluates attack-wise subsets rather than unified 20-class softmax diagnosis; pruning, quantization, and footprint are not reported.
Zeng and Mi [11]	2025	VeReMi Extension	Robust LSTM + FedAware FL	Aggregated accuracy only	Single/NS	NR	• Robust LSTM with FedAware aggregation for privacy-preserving non-IID IoV intrusion detection. • Reports mainly aggregate accuracy, without macro-F1, per-class analysis, dual observability, or embedded compression evidence.
Luo et al. [56]	2025	SynCAN/ ROAD	SR-LiCNN with spectral residuals + lightweight depthwise separable CNN	Binary	NA (in-vehicle signals)	Yes	• SR-LiCNN converts in-vehicle signal sequences into spectral-residual saliency images and uses lightweight depthwise separable CNNs. • Reports strong F1, 111,786 parameters, 5.776 MB memory, and millisecond inference, but targets binary intra-vehicle anomaly detection, not BSM-centric 20-class diagnosis.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Uddin et al. [24]	2025	CIC-IoV2024/ CIC-DDoS2019	Hierarchical ML IDS + Boruta	Multi	NA (CAN-network traffic)	Partially	• Three-level hierarchical IDS with Boruta/RF feature selection, distributed across vehicle, RSU, near-edge, and cloud layers. • Evaluated on CIC-IoV2024/ CIC-DDoS2019, not BSM-centric VeReMi Extension; no dual observability or unified 20-class BSM diagnosis.
Cao et al. [25]	2025	VeReMi Extension	Split learning + Llama3.1-8B	Binary anomaly	Single/NS	Edge-cloud split	• Edge-cloud split-learning IDS with lightweight vehicle-side tokenizer/classifier and cloud-side Llama3.1-8B backbone; reports 568 KB vehicle-side model and 50.22 ms detection time. • Not fully OBU-only; no unified 20-class diagnosis or ideal-versus-realistic BSM observability.
Ahsan et al. [45]	2025	VeReMi	RF + CatBoost + LR stacking with SHAP	Multi	Single/NS	NR	• Explainable SD-VANET ensemble using RF, CatBoost, LR stacking, and SHAP, reporting 99.68% accuracy and 99.45% F1-score on VeReMi. • Does not address VeReMi Extension, temporal sequence modeling, dual BSM observability, or compressed TFLite deployment.
Almansour et al. [53]	2025	CIC-IDS2017/ CSE-CIC-IDS2018/ Car-Hacking/ CAN-Train-Test	APFed + LDwCBN	Multi/ binary depending on dataset	NA (CAN-network traffic)	Partially	• APFed combines adaptive personalized FL, dynamic weight fusion, and LDwCBN for non-IID IoV data and resource-constrained devices. • Improves over FedAvg/FedProx, but does not evaluate BSM-centric VeReMi Extension, dual observability, or TFLite 20-class deployment.

Table 1. *Cont.*

Reference	Year	Dataset	Core Methodology	Classes	Observability	Edge Evidence	Critical Remarks
Mahesh et al. [46]	2025	VeReMi Extension	HyperLSTM + overlapping windows	Binary	Single/NS	Partially	• Proposes a HyperLSTM anomaly detector for VeReMi Extension with 128 LSTM units, dropout regularization, and overlapping temporal windows. • Reports 98% accuracy, 97.41% macro-F1, and 0.0006 s prediction latency for binary normal/attack detection. • Does not address unified 20-class diagnosis, ideal-versus-realistic BSM observability, sender-level leakage prevention, or compressed TensorFlow Lite deployment.
Jeyaram et al. [27]	2026	VANET-Malicious	Liquid Neural Network (LNN)	Binary	Ideal	Partially	• Uses Liquid Neural Networks with multipoint spiral search optimization for lightweight binary VANET attack detection. • Dynamic sequence tracking is promising, but concrete compression, serialized footprint, and execution-latency evidence are missing.
Proposed Work	2026	VeReMi Extension	Physics-informed LSTM + TFLite	Multi (20)	Ideal vs. realistic BSM observability	Yes	• Unified 20-class VeReMi Extension diagnosis under sender-level splitting. • Explicit comparison between clean simulator-side fields and noisy BSM-derived observables. • Compact 20-feature physics-informed representation without SMOTE/GAN balancing. • Pruning + TensorFlow Lite deployment artifact reaches 88.38 KB, 98.67% realistic macro-F1, 0.1283 ms realistic controlled-runtime latency, and 0.1398 ms ideal controlled-runtime latency.

The comparison confirms that existing VANET/IoV IDSs usually optimize only part of the problem. Some studies are BSM-oriented but binary, attack-wise, replay-focused, or not deployment-ready; others are edge-aware or federated, but they rely on CAN, network-traffic, cloud-assisted, or split-learning settings. The proposed work complements these efforts by combining 20-class VeReMi Extension diagnosis, dual BSM observability, sender-level leakage prevention, temporal-window analysis, physics-informed feature design, and a compact standalone TensorFlow Lite deployment artifact. A detailed quantitative comparison with the closest recent VeReMi-oriented IDSs is provided in Section 5.4.

3. Materials and Methods

This section describes the methodology used to construct, evaluate, and compress the proposed temporal-window-aware physics-informed edge-oriented IDS. The pipeline comprises seven stages: dataset preparation under two BSM observability regimes, leakage-aware sender-level partitioning, physics-informed feature engineering, robust normalization and sender-local temporal windowing, compact LSTM-based 20-class classification, edge-oriented TensorFlow Lite export, and temporal-window sensitivity analysis. The methodology jointly addresses diagnostic granularity, physical consistency, evaluation validity, temporal-context selection, and edge-deployment potential.

The two observability regimes are derived entirely from VeReMi Extension. The ideal regime uses clean simulator-side kinematic fields as an upper-bound diagnostic reference. The simulation-based realistic regime uses noise-inclusive BSM-derived fields reconstructed from the corresponding noise-free variables and dataset-provided sensor-error components. It is retained as the principal noise-robustness reference because it better imitates receiver-side measurement uncertainty than the ideal regime; nevertheless, it is not real-world BSM data. The temporal horizon is evaluated over $T \in \{10, 20, 30, 40, 50\}$, and the retained configuration is selected according to the observed robustness–latency trade-off.

3.1. Dataset and BSM Observability Regimes

The underlying simulation benchmark is VeReMi Extension [28], which extends the original VeReMi dataset [2]. More precisely, the experiments reported in this study use the publicly released labeled tabular derivative distributed as `mixalldata_clean.tar.xz` in Version 1 of the Mendeley Data record *VeReMi_Extension: Dataset for Misbehaviors in VANETs*, https://doi.org/10.17632/k62n4z9gdz.1 [57]. The archive contains the prepared file `mixalldata_clean.csv`, comprising 3,194,808 labeled BSM records and 30 columns. This tabular derivative was produced by combining the 39 VeReMi Extension datasets and mapping their ground-truth information to the receiver log records in order to add the categorical target attribute *class* [42,43]. The original class labels from 0 to 19 were preserved in the present study; the subsequent processing only constructs the ideal and realistic observability views, the physics-informed descriptors, and sender-local temporal sequences. Its structure has also been used in prior C-ITS misbehavior classification studies, where the 24 hourly ground-truth files were processed using fields such as sender identifier, pseudonym identifier, message identifier, send time, position, velocity, acceleration, heading, and their associated noise components [47]. It is particularly suitable for BSM-based IoV intrusion detection because it preserves the spatio-temporal semantics of inter-vehicular communication rather than reducing the task to generic network traffic or intra-vehicle CAN traces [29]. Recent dataset-oriented work also confirms the central role of VeReMi-style benchmarks for VANET misbehavior detection, while noting that DDoS-oriented scenarios remain only partially covered; Zhong et al. [58] therefore proposed a dedicated VANET DDoS dataset with constant-rate, increasing-rate, pulse, and increasing-pulse BSM flooding attacks. In this work, VeReMi Extension behaviors are encoded as a 20-class sequence-level classification problem, where class 0 denotes normal behavior and classes 1–19 denote attack or anomalous categories. The complete behavior taxonomy is reported in Table 2.

In the present work, both observability regimes are derived from the same VeReMi Extension simulated messages, senders, pseudonyms, timestamps, and class labels. The sole difference between the two regimes is whether the dataset-provided sensor-error components are included in the observable kinematic state. The *ideal* regime uses the noise-free component q_i. The *simulation-based realistic (noise-inclusive)* regime reconstructs the corresponding observable as $q_i^r = q_i + q_i^n$, where q_i^n is the sensor-error value supplied separately

by VeReMi Extension. Accordingly, "realistic regime" is used hereafter as a short label for this controlled noise-inclusive condition within the simulation benchmark. It indicates improved receiver-side measurement realism relative to clean simulator fields, but it does not refer to BSMs collected from physical vehicles, OBUs, RSUs, or operational V2X links.

Table 2. Detailed Classification of Behavior Types (0–19) in the VeReMi Extension Dataset.

Label	Behavior/Attack Name	Signification
0	**Normal**	Genuine BSMs reflecting real vehicular kinematics without any manipulation.
1	Constant Position	Transmits a fixed, static GPS position regardless of the vehicle's actual movement.
2	Constant Position Offset	Transmits the real position shifted by a fixed, pre-determined distance vector.
3	Random Position	Transmits completely random coordinates within the simulation map boundaries.
4	Random Position Offset	Transmits the real position shifted by a randomly generated distance vector.
5	Constant Speed	Transmits a fixed, unchanging speed value.
6	Constant Speed Offset	Transmits the real speed shifted by a fixed scalar value.
7	Random Speed	Transmits completely random, physically implausible speed values.
8	Random Speed Offset	Transmits the real speed shifted by a randomly generated scalar value.
9	Eventual Stop	The vehicle initially reports normal motion and subsequently transmits a stationary or stopped state while its actual motion continues.
10	Disruptive Attack	Transmits plausible-looking but unsafe kinematic information derived from other observed vehicles, potentially inducing inappropriate reactions.
11	Data Replay	Retransmits previously observed legitimate BSM content instead of the sender's current state.
12	Delayed Messages (Stale Messages)	Transmits buffered, temporally stale BSM states rather than the sender's current kinematic state.
13	DoS (Denial of Service)	Increases the BSM transmission rate to overload the communication channel.
14	DoS Random	Combines high-rate transmission with randomly generated kinematic values.
15	DoS Disruptive	Combines high-rate transmission with disruptive false kinematic values.
16	Traffic Congestion/ Grid Sybil	Uses several forged pseudonyms and spatially arranged fabricated vehicle states to create false traffic occupancy or congestion information.
17	Data Replay Sybil	Replays previously observed BSM content while using forged pseudonyms.
18	DoS Random Sybil	Combines high-rate transmission, rapidly changing forged pseudonyms, and randomly generated kinematic values.
19	DoS Disruptive Sybil	Combines high-rate transmission and rapidly changing forged pseudonyms with disruptive kinematic information.

3.1.1. Ideal BSM Observability

The *ideal BSM observability* regime uses clean simulator-side kinematic fields from VeReMi Extension. For each BSM emitted by a sender vehicle, the retained kinematic state is defined as

$$\mathbf{s}_i^{\text{ideal}} = [\texttt{posx}_i, \texttt{posy}_i, \texttt{spdx}_i, \texttt{spdy}_i, \texttt{aclx}_i, \texttt{acly}_i, \texttt{hedx}_i, \texttt{hedy}_i],$$

where position, speed, acceleration, and heading components correspond to clean ground-truth values. These fields are combined with contextual BSM attributes such as sender, pseudonym, message identifier, timestamp, and class label. The ideal input view is therefore based on:

$$\{\texttt{sender}, \texttt{senderPseudo}, \texttt{messageID}, \texttt{sendTime}, \texttt{class}, \texttt{posx}, \texttt{posy}, \texttt{spdx}, \texttt{spdy}, \texttt{aclx}, \texttt{acly}, \texttt{hedx}, \texttt{hedy}\}.$$

This ideal regime is not a deployment assumption; it provides an upper-bound reference for quantifying how discriminative the proposed physics-informed temporal representation is when the dataset-provided sensor-error components are not included in the observable kinematic state.

3.1.2. Realistic BSM Observability

The *realistic BSM observability* regime is also derived entirely from the VeReMi Extension simulation. Unlike the ideal regime, it incorporates the sensor-error component stored separately by the dataset rather than exposing only the noise-free kinematic component. For a generic kinematic variable

$$q \in \{\texttt{posx}, \texttt{posy}, \texttt{spdx}, \texttt{spdy}, \texttt{aclx}, \texttt{acly}, \texttt{hedx}, \texttt{hedy}\},$$

VeReMi Extension provides the noise-free value q_i and its associated sensor-error value q_i^n. The noise-inclusive observable used by the realistic regime is reconstructed as

$$q_i^r = q_i + q_i^n. \tag{1}$$

For example,

$$\texttt{posx_r}_i = \texttt{posx}_i + \texttt{posx_n}_i, \qquad \texttt{spdx_r}_i = \texttt{spdx}_i + \texttt{spdx_n}_i.$$

The resulting realistic kinematic state is

$$\mathbf{s}_i^{(\text{realistic})} = [\texttt{posx_r}_i, \texttt{posy_r}_i, \texttt{spdx_r}_i, \texttt{spdy_r}_i, \texttt{aclx_r}_i, \texttt{acly_r}_i, \texttt{hedx_r}_i, \texttt{hedy_r}_i]^\top.$$

The noise components `posx_n`, `posy_n`, and their speed, acceleration, and heading counterparts are used only to reconstruct the noise-inclusive observables. They are not supplied to the classifier as separate predictive features. Therefore, the model cannot directly observe or subtract the injected sensor error.

The term *realistic* is used in a relative benchmark sense: this regime is more representative of receiver-side observability than the ideal noise-free regime because it includes the sensor-error values supplied by VeReMi Extension. Nevertheless, it remains simulation-based and should not be interpreted as measurements collected from physical vehicles, OBUs, RSUs, or an over-the-air V2X deployment.

Table 3 summarizes the key characteristics and experimental roles of the two BSM observability regimes evaluated in this study.

Both observability regimes are processed through the same downstream pipeline: sender-level partitioning, temporal ordering, feature engineering, robust normalization,

LSTM architecture, compression variants, and evaluation metrics. This methodological symmetry ensures that performance differences between ideal and realistic observability are attributable to the observability condition rather than to changes in data partitioning, model design, or optimization. The original class distribution is preserved in both regimes, and no synthetic balancing is applied during dataset preparation. This choice avoids potentially kinematically inconsistent synthetic samples [30,33,34] and instead relies on physics-informed descriptors to expose motion inconsistency, temporal irregularity, replay behavior, pseudonym changes, and long-horizon constant-motion patterns before temporal learning.

Table 3. Summary of the two BSM observability regimes used in the proposed evaluation protocol.

Regime	Kinematic Fields Used	Role in the Experimental Protocol
Ideal BSM observability	`posx`, `posy`, `spdx`, `spdy`, `aclx`, `acly`, `hedx`, `hedy`	Upper-bound simulator-side evaluation using clean ground-truth kinematic variables. This regime measures the intrinsic discriminative capacity of the physics-informed IDS when motion fields are not corrupted by measurement noise.
Realistic BSM observability	`posx_r`, `posy_r`, `spdx_r`, `spdy_r`, `aclx_r`, `acly_r`, `hedx_r`, `hedy_r`	Noise-inclusive benchmark evaluation. Each observable is reconstructed from the two components supplied separately by VeReMi Extension: $q_i^r = q_i + q_i^n$. This regime imitates receiver-side sensor uncertainty but remains simulation-based.

3.2. Problem Formulation and Design Objectives

The objective of this work is to design a vehicular Intrusion Detection System (IDS) that jointly addresses six requirements: fine-grained multi-class diagnosis, explicit evaluation under different BSM observability assumptions, physical consistency with vehicular motion, leakage-aware temporal generalization, measurable edge-deployment potential, and reproducible compression-aware optimization. In the Internet of Vehicles (IoV), vehicles exchange Basic Safety Messages (BSMs) containing kinematic and contextual information such as position, speed, acceleration, heading, transmission timing, message identifiers, and pseudonym-related identifiers [7,8]. The detection problem is formulated over sender-specific BSM streams and instantiated under the two simulation-based regimes defined in Section 3.1: ideal noise-free observability and realistic noise-inclusive observability.

Let

$$\mathcal{O} = \{\text{ideal}, \text{realistic}\}$$

denote the set of observability regimes. For a sender vehicle v, let

$$\mathcal{M}^{(v,o)} = \{m_1^{(v,o)}, m_2^{(v,o)}, \ldots, m_{n_v}^{(v,o)}\}, \qquad o \in \mathcal{O},$$

be the temporally ordered sequence of BSMs associated with that sender under regime o. Each message $m_i^{(v,o)}$ is observed at time $t_i^{(v)}$, contains contextual fields such as sender, senderPseudo, and messageID, and carries a class label

$$y_i^{(v)} \in \{0, 1, \ldots, C-1\},$$

where $C = 20$ corresponds to one normal class and 19 attack or anomalous classes in VeReMi Extension [28,29]. The label is shared across observability regimes; only the kinematic fields made available to the detector differ.

Under ideal observability, the kinematic state is derived from clean simulator-side fields:

$$\mathbf{s}_i^{(v,\text{ideal})} = [posx_i, posy_i, spdx_i, spdy_i, aclx_i, acly_i, hedx_i, hedy_i]^\top.$$

Under realistic observability, the corresponding state is derived from noisy BSM-like fields:

$$\mathbf{s}_i^{(v,\text{realistic})} = [posx_r_i, posy_r_i, spdx_r_i, spdy_r_i, aclx_r_i, acly_r_i, hedx_r_i, hedy_r_i]^\top.$$

Thus, the two regimes define controlled input views of the same sender histories, timestamps, pseudonyms, message identifiers, and labels.

The learning objective is to estimate a sequence-level decision function

$$f_\theta^{(o,T)} : \mathbb{R}^{T\times d} \to \{0, 1, \ldots, C-1\},$$

where T is the temporal window length, $d = 20$ is the number of physics-informed descriptors, and θ denotes the model parameters. For each sender v, temporal windows are constructed as

$$\mathbf{X}_i^{(v,o,T)} = \left[\mathbf{x}_{i-T}^{(v,o)}, \mathbf{x}_{i-T+1}^{(v,o)}, \ldots, \mathbf{x}_{i-1}^{(v,o)}\right] \in \mathbb{R}^{T\times d}, \tag{2}$$

where $\mathbf{x}_j^{(v,o)} \in \mathbb{R}^d$ is the engineered d-dimensional feature vector extracted from message $m_j^{(v,o)}$. The target is the label of the next message,

$$\hat{y}_i^{(v,o,T)} = f_\theta^{(o,T)}\left(\mathbf{X}_i^{(v,o,T)}\right), \qquad y_i^{(v)} = \text{label}(m_i^{(v,o)}).$$

This one-step-ahead formulation uses recent sender-local behavior to classify the next incoming BSM, preventing windows from crossing sender boundaries or train-validation-test partitions.

For a sender v whose chronologically ordered trajectory contains n_v BSMs, the first T messages provide the historical context required to construct the first labeled sequence. Therefore, the number of supervised temporal windows generated for sender v is

$$N_v(T) = \max(n_v - T, 0). \tag{3}$$

If $n_v \leq T$, the sender does not provide a complete temporal history and consequently generates no supervised sequence. If $n_v > T$, each additional message after the first T messages defines one input window and its corresponding next-message target. Thus, increasing T provides more temporal context per sequence but reduces the number of available supervised samples.

Table 4 summarizes the mathematical symbols and notation used throughout the problem formulation and evaluation protocol.

The problem is challenging because it combines heterogeneous 20-class behavior, severe class imbalance, physical motion constraints, edge-inference requirements, and observability uncertainty. In VeReMi Extension, normal traffic and a few dominant attack classes account for most observations, while several safety-relevant classes remain rare [28,29]. At the same time, BSM streams must satisfy temporal continuity, displacement-speed consistency, bounded acceleration variation, and coherent heading evolution. Since the

target IDS is intended for embedded inference, high accuracy is insufficient without a small footprint and very low latency [9,19,20]. Finally, evaluating only clean ground-truth observability can overestimate deployment-time performance, which motivates the explicit realistic-observability regime.

Table 4. Main notation used in the problem formulation and evaluation protocol.

Symbol	Meaning
$\mathcal{O}$	Set of simulation-based observability regimes, with $\mathcal{O} = \{\text{ideal}, \text{realistic}\}$; "realistic" denotes the noise-inclusive condition defined in Section 3.1.
o	Observability regime used to construct the input view.
v	Sender vehicle identifier.
$\mathcal{M}^{(v,o)}$	Temporally ordered BSM stream associated with sender v under observability regime o.
$m_i^{(v,o)}$	i-th BSM of sender v under regime o.
$y_i^{(v)}$	Ground-truth class label of the i-th BSM.
C	Number of behavior classes; in this work, $C = 20$.
$\mathbf{x}_i^{(v,o)}$	Physics-informed d-dimensional feature vector extracted from message $m_i^{(v,o)}$.
d	Feature dimensionality; in this work, $d = 20$.
T	Temporal window length in BSMs, with $T \in \{10, 20, 30, 40, 50\}$.
$\mathbf{X}_i^{(v,o,T)}$	Sender-local temporal input window of size $T \times d$.
$f_\theta^{(o,T)}$	LSTM-based classifier parameterized by θ for regime o and temporal window T.

The formulation is guided by six design requirements:

- **R1** requires dual-observability validity: the IDS should be evaluated under both ideal and realistic BSM observability, using the same class labels, temporal-window candidates, model family, and evaluation logic so that the observed performance gap reflects the impact of observability rather than methodological confounders.
- **R2** requires fine-grained diagnosis across the 20 VeReMi Extension behavior classes, since replay, Sybil-based attacks, disruptive falsification, and flooding behaviors may require different mitigation, trust-management, or response strategies rather than the same generic malicious alarm [6,10,18].
- **R3** requires a physics-informed and non-privileged representation: the model should explicitly encode temporal, kinematic, and behavioral consistency for replay, offset, disruptive, random-motion, and eventual-stop behaviors, while the realistic pipeline must use only observable noisy fields and not simulator noise terms as predictive inputs [33–35].
- **R4** requires leakage-aware temporal generalization through sender-disjoint train, validation, and test partitions, because consecutive BSMs from the same sender are strongly correlated and random message-level splitting may overestimate generalization.
- **R5** requires edge deployability in terms of serialized footprint and inference latency, ruling out otherwise accurate but overly heavy ensemble, transformer, or infrastructure-dependent solutions whose memory and runtime costs are difficult to reconcile with constrained vehicular hardware [13,14,24,25].

- **R6** requires a reproducible compression-aware optimization path from the dense baseline to the exported deployment-oriented model. In this work, deployment feasibility is assessed through four variants: dense Keras, compression-only TensorFlow Lite, pruning-only stripped Keras, and pruning-plus-compression TensorFlow Lite. This protocol separates the effect of TensorFlow Lite conversion from pruning and avoids treating the final compressed model as a black-box optimization. The compression strategy is intentionally simpler than complex teacher–student or multi-stage compression pipelines, which may introduce substantial implementation overhead in deployment-oriented vehicular IDS settings [21,22,36].

Guided by these requirements, the proposed framework constructs two observation-specific input views from VeReMi Extension, extracts a compact 20-dimensional physics-informed feature space from sender-specific histories, and evaluates temporal windows with $T \in \{10, 20, 30, 40, 50\}$. The same lightweight LSTM classifier is used for 20-class diagnosis under each observability and temporal configuration, and its dense, TFLite, pruning-only, and pruning-plus-compression variants are compared in terms of diagnostic performance, temporal-context sensitivity, observability robustness, footprint, and inference latency. The temporal sensitivity analysis then retains $T = 40$ as the main deployment-oriented configuration because it provides the strongest robustness–latency compromise under realistic BSM observability while remaining close to the best ideal-observability performance.

3.3. Leakage-Aware Sender-Level Partitioning

A central requirement in BSM-based vehicular IDS is to prevent trajectory leakage between training, validation, and test data. Consecutive BSMs emitted by the same sender are strongly correlated in position, speed, acceleration, heading, timing, and pseudonym behavior. A random message-level split could therefore place adjacent fragments of the same trajectory in different partitions, allowing the model to memorize sender-specific motion patterns instead of learning attack-generalizable signatures. This risk is particularly important for sequential IDSs because the input representation explicitly exploits temporal continuity.

To avoid this issue, the proposed framework applies a strict sender-level partitioning protocol before feature engineering, scaling, or temporal window construction. Let

$$\mathcal{V} = \{v_1, v_2, \ldots, v_{N_v}\}$$

denote the set of unique sender identifiers in the VeReMi Extension data. The set of senders is first shuffled using a fixed random seed, $R = 42$, in order to ensure reproducibility. The shuffled sender set is then divided into a preliminary training pool and a test set according to an 80/20 sender-level split:

$$\mathcal{V}_{\text{pretrain}} \cup \mathcal{V}_{\text{test}} = \mathcal{V}, \qquad \mathcal{V}_{\text{pretrain}} \cap \mathcal{V}_{\text{test}} = \emptyset.$$

A validation set is subsequently extracted from the preliminary training pool by assigning 10% of the preliminary training senders to validation:

$$\mathcal{V}_{\text{train}} \cup \mathcal{V}_{\text{val}} = \mathcal{V}_{\text{pretrain}}, \qquad \mathcal{V}_{\text{train}} \cap \mathcal{V}_{\text{val}} = \emptyset.$$

This results in an effective split of approximately 72% training senders, 8% validation senders, and 20% test senders. All BSMs generated by a given sender are assigned to exactly one of these three partitions:

$$\mathcal{V}_{\text{train}} \cap \mathcal{V}_{\text{val}} = \mathcal{V}_{\text{train}} \cap \mathcal{V}_{\text{test}} = \mathcal{V}_{\text{val}} \cap \mathcal{V}_{\text{test}} = \emptyset.$$

For each observability regime $o \in \{\text{ideal}, \text{realistic}\}$, the corresponding message-level datasets are defined as

$$\mathcal{D}^{(o)}_{train} = \{m^{(v,o)}_i : v \in \mathcal{V}_{train}\},\ \mathcal{D}^{(o)}_{val} = \{m^{(v,o)}_i : v \in \mathcal{V}_{val}\},\ \text{and}\ \mathcal{D}^{(o)}_{test} = \{m^{(v,o)}_i : v \in \mathcal{V}_{test}\}.$$

The same sender partitions are reused for ideal and realistic BSM observability, ensuring that performance differences between the two regimes are caused by the observability condition rather than by different vehicles, trajectories, or label allocations.

In the implemented pipeline, the complete benchmark contains 3,194,808 BSM records. Sender-level partitioning produces 17,757 training senders, 1973 validation senders, and 4933 test senders. At the raw message level, these partitions correspond to 2,303,374 training messages, 253,372 validation messages, and 638,062 test messages, as summarized in Table 5.

After partitioning, all preprocessing operations are performed independently inside each split. Feature engineering is applied separately to training, validation, and test data, and all temporal operations are computed within sender-specific groups. This prevents rolling statistics, temporal differences, repeated-message indicators, pseudonym-change descriptors, and position-consistency residuals from being influenced by messages outside the corresponding split. The split is also not performed at the class-label level, and no synthetic rebalancing is applied during partitioning, preserving the natural trajectory-level structure of VeReMi Extension.

Table 5. Sender-level partitioning used in both ideal and realistic BSM observability regimes.

Partition	Unique Senders	Raw BSMs	Role
Training	17,757	2,303,374	Model fitting and scaler estimation
Validation	1973	253,372	Hyperparameter monitoring and early selection
Test	4933	638,062	Final held-out evaluation
Total	**24,663**	**3,194,808**	Complete benchmark

This leakage-aware protocol supports both the dual-observability design and the temporal-window sensitivity analysis. Because the ideal and realistic views share the same senders, timestamps, message identifiers, pseudonym context, and labels, the comparison between clean ground-truth observability and noisy BSM observability remains controlled. The same fixed sender partitions are also reused for all $T \in \{10, 20, 30, 40, 50\}$, ensuring that differences across temporal horizons reflect temporal-context length rather than changes in the train-validation-test split. The following subsection describes the physics-informed feature engineering stage applied after this sender-level partitioning step.

3.4. Physics-Informed Feature Engineering

Purely data-driven vehicular IDS models may learn empirical correlations without explicitly encoding the physical feasibility of motion. In BSM-centric detection, however, many attacks manifest as violations of temporal continuity, kinematic plausibility, message uniqueness, pseudonym behavior, or persistent motion patterns. A single BSM may therefore appear plausible when inspected in isolation, while becoming suspicious once it is

compared with the sender's recent trajectory. For this reason, the proposed framework uses sender-local physics-informed feature engineering rather than generic tabular expansion or synthetic augmentation.

The feature design follows three principles. First, all descriptors must be computable from information available to a receiver-side BSM monitor, under either ideal or realistic observability. Second, all temporal statistics must be sender-local and shifted, so that the descriptor of the current message depends only on previous messages from the same sender and never on future samples. Third, the same 20-feature representation must be kept fixed across all temporal horizons $T \in \{10, 20, 30, 40, 50\}$, ensuring that the temporal-window sensitivity analysis reflects the effect of sequence length rather than changes in the input representation.

Feature engineering is applied after sender-level partitioning and independently inside each split. All temporal differences, rolling statistics, repeated-message indicators, position-consistency residuals, pseudonym-change indicators, and long-horizon ratios are computed only within sender-specific trajectories already assigned to the training, validation, or test partition. This prevents preprocessing leakage across senders or across train–validation–test boundaries while preserving the chronological semantics of each BSM stream.

For compactness and consistency, the following notation is used throughout this subsection. The symbol v written in italic denotes the sender identifier, i denotes the chronological index of a BSM within that sender's trajectory, and $o \in \{\text{ideal}, \text{realistic}\}$ denotes the observability regime. To avoid overloading the equations, the sender index is omitted from the following kinematic-vector notation; all quantities are implicitly computed within the trajectory of the same sender.

For the i-th BSM, let

$$\mathbf{p}_i^{(o)} = \begin{bmatrix} p_{x,i}^{(o)} \\ p_{y,i}^{(o)} \end{bmatrix}, \qquad \mathbf{v}_i^{(o)} = \begin{bmatrix} v_{x,i}^{(o)} \\ v_{y,i}^{(o)} \end{bmatrix}, \qquad \mathbf{a}_i^{(o)} = \begin{bmatrix} a_{x,i}^{(o)} \\ a_{y,i}^{(o)} \end{bmatrix}, \qquad \mathbf{h}_i^{(o)} = \begin{bmatrix} h_{x,i}^{(o)} \\ h_{y,i}^{(o)} \end{bmatrix}.$$

Here, $\mathbf{p}_i^{(o)}$ is the two-dimensional position vector, $\mathbf{v}_i^{(o)}$ is the two-dimensional velocity vector, $\mathbf{a}_i^{(o)}$ is the two-dimensional acceleration vector, and $\mathbf{h}_i^{(o)}$ is the two-dimensional heading or motion-direction vector. Their components correspond respectively to the BSM fields `posx/posy`, `spdx/spdy`, `aclx/acly`, and `hedx/hedy`, or to their reconstructed noise-inclusive counterparts under realistic observability. Position is expressed in metres, velocity in metres per second, and acceleration in metres per second squared. The superscript $^\top$, when used with a row representation, denotes vector transposition and is unrelated to the temporal-window length T.

The scalar speed magnitude is defined using the Euclidean, or ℓ_2, norm of the velocity vector:

$$u_i^{(o)} = \left\| \mathbf{v}_i^{(o)} \right\|_2 = \sqrt{\left(v_{x,i}^{(o)}\right)^2 + \left(v_{y,i}^{(o)}\right)^2}. \tag{4}$$

The subscript 2 in $\| \cdot \|_2$ identifies the Euclidean norm; it is not a message index, a time index, or the square of the vector.

The sender-local inter-message interval is

$$\Delta t_i = t_i - t_{i-1},$$

where t_i is the transmission time of the i-th BSM. In addition,

$$\mathcal{W}_{i,k} = \begin{cases} \{\max(1, i-k), \ldots, i-1\}, & i \geq 2, \\ \emptyset, & i = 1, \end{cases}$$

where $\mathcal{W}_{i,k}$ is the shifted sender-local history containing at most the previous k messages. The indicator function $\mathbb{I}[\cdot]$ equals 1 when its condition is true and 0 otherwise. When a shifted rolling statistic is undefined because the history is empty or contains insufficient samples, its value is initialized to zero, consistently with the implemented missing-value handling. All differences and rolling statistics are computed independently within one sender trajectory and one data partition.

For each sender v, messages are first sorted by transmission time. Using the vectors defined above, the corresponding eight-dimensional instantaneous kinematic state is written as

$$\mathbf{s}_i^{(v,o)} = \left[(\mathbf{p}_i^{(o)})^\top, (\mathbf{v}_i^{(o)})^\top, (\mathbf{a}_i^{(o)})^\top, (\mathbf{h}_i^{(o)})^\top \right]^\top.$$

Under ideal observability, this state contains the noise-free kinematic components. Under realistic observability, it contains the reconstructed noise-inclusive components defined in Equation (1).

The first descriptor group retains the instantaneous two-dimensional kinematic state $\mathbf{s}_i^{(v,o)}$, which provides the position, speed, acceleration, and heading components used by the downstream residual and motion-consistency descriptors. The second group captures timing and replay behavior. The retained inter-message-delay descriptor is

$$\texttt{dt}_i = \max(\Delta t_i, 10^{-4}).$$

The first message of each sender trajectory is initialized to 0.1 s.

Replay-related behavior is captured through a sender-local repeated-message indicator:

$$\texttt{msg_repeated}_i = \mathbb{I}[\texttt{messageID}_i \in \{\texttt{messageID}_1, \ldots, \texttt{messageID}_{i-1}\}],$$

where $\texttt{messageID}_i$ denotes the message identifier of the i-th BSM. The feature is equal to one if the current message identifier has already appeared earlier in the same sender trajectory, and zero otherwise. These two descriptors target timing irregularities, flooding-like behavior, and replay-related message reuse. They are particularly relevant for Data Replay and replay-based Sybil variants, where retransmitted messages may preserve valid-looking kinematic values while violating temporal uniqueness.

Using the speed magnitude defined in Equation (4), a short-horizon speed history is obtained from a shifted rolling mean over the previous five speed magnitudes:

$$\overline{u}_{i,-}^{(5,o)} = \frac{1}{|\mathcal{W}_{i,5}|} \sum_{j \in \mathcal{W}_{i,5}} u_j^{(o)},$$

The eventual-stop descriptor is defined as

$$\texttt{eventual_stop_score}_i = \max\left(\overline{u}_{i,-}^{(5,o)} - u_i^{(o)}, 0 \right).$$

This feature gives a compact measure of abrupt or progressive reported deceleration relative to the recent speed history. It is designed to expose stop-related contradictions, especially in eventual-stop scenarios where malicious behavior emerges over time rather than from a single instantaneous value.

Speed persistence is encoded through the absolute change in the sender-local speed magnitude between two consecutive BSMs:

$$\texttt{speed_change}_i = \left| u_i^{(o)} - u_{i-1}^{(o)} \right|. \tag{5}$$

The quantity in Equation (5) is expressed in $\mathrm{m\,s^{-1}}$. It measures a change in speed magnitude and must not be confused with the consecutive-position displacement $d_i^{(o)}$, expressed in metres, that is subsequently used for constant-position detection.

A nearly constant-speed transition is identified as

$$\texttt{constant_speed_flag}_i = \mathbb{I}[\texttt{speed_change}_i \leq \tau_v], \tag{6}$$

where τ_v is a near-zero speed-change tolerance expressed in $\mathrm{m\,s^{-1}}$. The nominal value is $\tau_v = 0.025\ \mathrm{m\,s^{-1}}$. This tolerance avoids defining speed persistence through exact floating-point equality while remaining small relative to ordinary vehicle-speed variations.

The local sensitivity of this tolerance is evaluated over

$$\mathcal{T}_v = \{0.020, 0.025, 0.030\}\ \mathrm{m\,s^{-1}}. \tag{7}$$

Only τ_v is varied in this analysis. The realistic constant-position threshold remains fixed at $\tau_p^{(\mathrm{realistic})} = 4.5$ m. All candidates use the same sender-level partition, random seed, $T = 40$ temporal configuration, 20-feature representation, and 64/32/32 LSTM architecture. The comparison is conducted exclusively on the validation partition; the held-out test partition is not used for threshold assessment.

As shown in Table 6, the three candidates lie on a narrow validation plateau. The value $\tau_v = 0.030\ \mathrm{m\,s^{-1}}$ yields a marginally higher accuracy than $0.025\ \mathrm{m\,s^{-1}}$, by approximately 0.001 percentage points, corresponding to only two additional correctly classified validation sequences out of 178,184. In contrast, $\tau_v = 0.025\ \mathrm{m\,s^{-1}}$ provides the highest validation weighted-F1 and macro-F1, while also outperforming $0.020\ \mathrm{m\,s^{-1}}$ on all three metrics. We therefore retain $\tau_v = 0.025\ \mathrm{m\,s^{-1}}$ as the class-aware operating point within this local plateau. This value is a benchmark-specific near-zero speed-change tolerance rather than a universal physical threshold.

Table 6. Local validation sensitivity of the dense realistic-observability model to the near-zero speed-change tolerance at $T = 40$, with $\tau_p^{(\mathrm{realistic})} = 4.5$ m fixed.

τ_v ($\mathrm{m\,s^{-1}}$)	Validation Accuracy (%)	Validation Weighted-F1 (%)	Validation Macro-F1 (%)
0.020	99.329	99.315	98.736
0.025	**99.350**	**99.338**	**98.779**
0.030	99.351	99.332	98.763

Instead of retaining the binary flag directly, the final representation uses the shifted sender-local ratio

$$\texttt{constant_speed_ratio_roll_100}_i = \frac{1}{|\mathcal{W}_{i,100}|} \sum_{j \in \mathcal{W}_{i,100}} \texttt{constant_speed_flag}_j. \tag{8}$$

This descriptor characterizes whether a sender has reported a nearly constant speed over a long recent horizon and is intended to expose persistent constant-speed and speed-manipulation behaviors.

To capture disruptive motion patterns, the framework derives a compact composite score from acceleration variation, heading variation, and braking intensity. The acceleration variation is computed as

$$\Delta a_i^{(o)} = \sqrt{\left(a_{x,i}^{(o)} - a_{x,i-1}^{(o)}\right)^2 + \left(a_{y,i}^{(o)} - a_{y,i-1}^{(o)}\right)^2},$$

and the corresponding jerk-like quantity is

$$\texttt{jerk}_i = \frac{\Delta a_i^{(o)}}{\texttt{dt}_i + \epsilon}, \qquad \epsilon = 10^{-6}.$$

Heading variation is computed as

$$\texttt{heading_change}_i = \left|h_{x,i}^{(o)} - h_{x,i-1}^{(o)}\right| + \left|h_{y,i}^{(o)} - h_{y,i-1}^{(o)}\right|.$$

The braking component is defined as

$$\texttt{braking_score}_i = \max\left(\frac{u_{i-1}^{(o)} - u_i^{(o)}}{\texttt{dt}_i + \epsilon}, 0\right).$$

The final retained disruptive descriptor is then

$$\texttt{disruptive_score}_i = \texttt{braking_score}_i + \texttt{jerk}_i + \texttt{heading_change}_i.$$

This feature aggregates abrupt deceleration, acceleration instability, and heading discontinuity into a single physically interpretable descriptor. It is particularly relevant for disruptive manipulation attacks, which may report plausible-looking values while inducing unsafe reactions in neighboring vehicles.

Identity-context behavior is represented through the pseudonym-change indicator

$$\texttt{pseudo_change}_i = \mathbb{I}[\texttt{senderPseudo}_i \neq \texttt{senderPseudo}_{i-1}],$$

with the first message of each sender trajectory set to zero. This descriptor provides a low-cost indicator of pseudonym dynamics and helps expose Sybil-related or identity-manipulation behaviors when combined with the temporal and kinematic descriptors.

Position consistency is encoded by comparing the observed current position with the position predicted from the previous position and previous speed. For each sender, the expected position under a constant-velocity short-step approximation is

$$\widehat{p}_{x,i}^{(o)} = p_{x,i-1}^{(o)} + v_{x,i-1}^{(o)}\texttt{dt}_i, \qquad \widehat{p}_{y,i}^{(o)} = p_{y,i-1}^{(o)} + v_{y,i-1}^{(o)}\texttt{dt}_i.$$

The resulting position-consistency residual is

$$\texttt{pos_error}_i = \sqrt{\left(p_{x,i}^{(o)} - \widehat{p}_{x,i}^{(o)}\right)^2 + \left(p_{y,i}^{(o)} - \widehat{p}_{y,i}^{(o)}\right)^2}.$$

To capture local instability in this residual, the framework also retains a shifted rolling standard deviation:

$$\texttt{pos_jitter}_i = \mathrm{std}\left(\texttt{pos_error}_j : j \in \mathcal{W}_{i,5}\right),$$

computed over the previous five sender-local residuals, with missing values subsequently replaced by zero. These two features are designed to expose position falsification, offset, random-position, and disruptive behaviors that violate displacement-speed consistency.

Zero-pattern behavior is captured from the eight current kinematic components. Let

$$\boldsymbol{\kappa}_i^{(o)} = \left[p_{x,i}^{(o)}, p_{y,i}^{(o)}, v_{x,i}^{(o)}, v_{y,i}^{(o)}, a_{x,i}^{(o)}, a_{y,i}^{(o)}, h_{x,i}^{(o)}, h_{y,i}^{(o)}\right]^\top \in \mathbb{R}^8$$

denote the ordered kinematic-component vector. Its zero-count descriptor is

$$\texttt{zero_count_2d}_i = \sum_{\ell=1}^{8} \mathbb{I}\left[\kappa_{i,\ell}^{(o)} = 0\right],$$

where ℓ indexes the eight kinematic components. The all-zero kinematic flag is

$$\texttt{all_kinematics_zero_2d}_i = \mathbb{I}[\texttt{zero_count_2d}_i = 8].$$

and the retained long-horizon zero-pattern feature is

$$\texttt{zero_ratio_roll_100}_i = \frac{1}{|\mathcal{W}_{i,100}|} \sum_{j \in \mathcal{W}_{i,100}} \texttt{all_kinematics_zero_2d}_j.$$

This feature is useful for detecting abnormal repeated zero-valued reports that may appear in stop-related, flooding-related, or malformed-motion behaviors.

Finally, constant-position behavior is encoded using the Euclidean displacement between consecutive sender-local positions:

$$d_i^{(o)} = \sqrt{\left(p_{x,i}^{(o)} - p_{x,i-1}^{(o)}\right)^2 + \left(p_{y,i}^{(o)} - p_{y,i-1}^{(o)}\right)^2}.$$

$$\texttt{constant_position_flag}_i^{(o)} = \mathbb{I}\left[d_i^{(o)} \leq \tau_p^{(o)}\right]. \tag{9}$$

The ideal-regime threshold is fixed to $\tau_p^{(\text{ideal})} = 0.025$ m, corresponding to a 2.5 cm near-zero displacement tolerance. This tolerance avoids relying on an exact floating-point equality when identifying consecutive constant-position reports in the noise-free kinematic fields. It is not intended to represent a positioning-error model.

For realistic observability, the position-error components supplied by VeReMi Extension substantially widen the distribution of consecutive observed displacements. Since $\tau_p^{(\text{realistic})}$ is a benchmark-calibration tolerance rather than a universal physical constant, a local sensitivity analysis was conducted over the interval

$$\mathcal{T}_p^{(\text{realistic})} = \{4.0, 4.5, 5.0\} \text{ m},$$

using a 0.5 m sampling step.

All candidates were evaluated using the same sender-disjoint partition, random seed, $T = 40$ temporal configuration, 20-feature representation, and dense LSTM architecture. Threshold selection was performed exclusively on the validation split using accuracy and weighted-F1 as the primary criteria, complemented by macro-F1 to account for minority-class behavior. The held-out test set was not consulted during threshold selection. Because the threshold changes only the computation of the constant-position descriptor while preserving the input dimensionality and model graph, all candidates have the same parameter count and theoretical inference complexity.

As shown in Table 7, $\tau_p^{(\text{realistic})} = 4.5$ m achieves the highest validation accuracy, weighted-F1, and macro-F1 within the investigated local interval. It is therefore retained for all subsequent realistic-observability experiments. The relatively small variation across the three candidates also indicates that the detector is not excessively sensitive to moderate perturbations of this tolerance. Nevertheless, 4.5 m should be interpreted as a benchmark-specific calibration value rather than as a universal positioning threshold for operational V2X systems.

Table 7. Local validation sensitivity of the dense realistic-observability model to the constant-position threshold at $T = 40$.

$\tau_p^{(\text{realistic})}$ (m)	Validation Accuracy (%)	Validation Weighted-F1 (%)	Validation Macro-F1 (%)
4.0	99.302	99.284	98.638
4.5	**99.337**	**99.323**	**98.756**
5.0	99.250	99.229	98.574

The retained feature is the shifted rolling ratio

$$\texttt{constant_position_ratio_roll_100}_i^{(o)} = \frac{1}{|\mathcal{W}_{i,100}|} \sum_{j \in \mathcal{W}_{i,100}} \texttt{constant_position_flag}_j^{(o)}. \quad (10)$$

This descriptor targets attacks that repeatedly report a fixed or nearly fixed position over time.

The eight retained feature families are not generic statistical descriptors; each family is designed to expose a specific physical or communication-level inconsistency associated with one or more VeReMi Extension attack groups. Instantaneous kinematics provide the observable motion state and remain necessary for position-, speed-, acceleration-, and heading-based falsification. Timing and replay cues target abnormal inter-message delays, flooding-like emission patterns, and message identifier reuse. Stop and speed-persistence features expose constant-speed, speed-offset, and eventual-stop behaviors whose signatures emerge over several BSMs. The disruptive-motion descriptor summarizes braking, jerk-like acceleration variation, and heading discontinuity, which are relevant to attacks that inject plausible but unsafe motion states. Pseudonym dynamics capture sender-identity evolution and are therefore directly linked to Sybil-related and identity-dependent misbehavior. Position-consistency descriptors compare observed displacement with motion predicted from previous velocity, making them physically aligned with position offset, random-position, and displacement-inconsistency attacks. Zero-pattern descriptors capture malformed or degenerate kinematic reports, while constant-position behavior targets fixed-position and stop-like anomalies. This design therefore maps each descriptor family to a physically meaningful BSM evidence channel rather than relying on purely data-driven feature expansion.

The final implemented representation contains 20 features. Therefore, $d = 20$ in both observability regimes. For ideal BSM observability, the retained feature vector is

$$\begin{aligned}\mathbf{x}_i^{(v,\text{ideal})} = [&\texttt{posx}, \texttt{posy}, \texttt{spdx}, \texttt{spdy}, \texttt{aclx}, \texttt{acly}, \texttt{hedx}, \texttt{hedy}, \texttt{dt}, \texttt{msg_repeated},\\ &\texttt{eventual_stop_score}, \texttt{disruptive_score}, \texttt{pseudo_change}, \texttt{pos_error}, \texttt{pos_jitter},\\ &\texttt{zero_count_2d}, \texttt{zero_ratio_roll_100}, \texttt{speed_change},\\ &\texttt{constant_speed_ratio_roll_100}, \texttt{constant_position_ratio_roll_100}]^\top. \quad (11)\end{aligned}$$

For realistic BSM observability, the same feature logic is retained, but the eight base kinematic fields are replaced by their noisy BSM-derived counterparts:

$$\mathbf{x}_i^{(v,\text{realistic})} = [\texttt{posx_r}, \texttt{posy_r}, \texttt{spdx_r}, \texttt{spdy_r}, \texttt{aclx_r}, \texttt{acly_r}, \texttt{hedx_r}, \texttt{hedy_r}, \texttt{dt}, \texttt{msg_repeated}, \texttt{eventual_stop_score}, \texttt{disruptive_score}, \texttt{pseudo_change}, \texttt{pos_error}, \texttt{pos_jitter}, \texttt{zero_count_2d}, \texttt{zero_ratio_roll_100}, \texttt{speed_change}, \texttt{constant_speed_ratio_roll_100}, \texttt{constant_position_ratio_roll_100}]^\top. \quad (12)$$

Table 8 consolidates the mathematical definitions, symbol interpretations, and operational roles of the 20 retained descriptors. The eight instantaneous kinematic components provide the common observable state, while the twelve engineered descriptors expose timing, replay, pseudonym, position-consistency, zero-pattern, speed-persistence, disruptive-motion, and constant-position behaviors that may remain plausible at a single timestamp but become suspicious over a sender-local history.

Table 8. Unified definitions of the 20 retained physics-informed features. All temporal operations are sender-local and use only the current or preceding messages.

Retained Feature(s)	Mathematical Definition and Interpretation
`posx`, `posy`, `spdx`, `spdy`, `aclx`, `acly`, `hedx`, `hedy`	The eight components of the instantaneous kinematic state $\mathbf{s}_i^{(o)} = [(\mathbf{p}_i^{(o)})^\top, (\mathbf{v}_i^{(o)})^\top, (\mathbf{a}_i^{(o)})^\top, (\mathbf{h}_i^{(o)})^\top]^\top$. The ideal regime uses the noise-free values q_i. The realistic regime uses the reconstructed values $q_i^r = q_i + q_i^n$.
`dt`	Sender-local inter-message interval: $\texttt{dt}_i = \max(t_i - t_{i-1}, 10^{-4})$. The first value of each sender trajectory is initialized to 0.1 s.
`msg_repeated`	Repeated-message indicator: $\texttt{msg_repeated}_i = \mathbb{I}[\texttt{messageID}_i \in \{\texttt{messageID}_1, \ldots, \texttt{messageID}_{i-1}\}]$.
`eventual_stop_score`	Recent deceleration relative to the shifted five-message speed history: $\texttt{eventual_stop_score}_i = \max\left(\frac{1}{\lvert\mathcal{W}_{i,5}\rvert}\sum_{j\in\mathcal{W}_{i,5}} u_j^{(o)} - u_i^{(o)}, 0\right)$.
`speed_change`	Absolute change in speed magnitude: $\texttt{speed_change}_i = \lvert u_i^{(o)} - u_{i-1}^{(o)}\rvert$.

Table 8. *Cont.*

Retained Feature(s)	Mathematical Definition and Interpretation
`constant_speed_ratio_roll_100`	Shifted proportion of nearly constant-speed transitions: $$\texttt{constant_speed_ratio_roll_100}_i = \frac{1}{\lvert\mathcal{W}_{i,100}\rvert} \sum_{j \in \mathcal{W}_{i,100}} \mathbb{I}\Big[\texttt{speed_change}_j \le \tau_v\Big],$$ $$\tau_v = 0.025\ \mathrm{m\,s^{-1}}.$$ The retained tolerance is supported by the local validation analysis in Table 6.
`disruptive_score`	Composite disruptive-motion descriptor: $$b_i = \max\left(\frac{u_{i-1}^{(o)} - u_i^{(o)}}{\texttt{dt}_i + \epsilon}, 0\right),$$ $$j_i = \frac{\lVert \mathbf{a}_i^{(o)} - \mathbf{a}_{i-1}^{(o)} \rVert_2}{\texttt{dt}_i + \epsilon}, \qquad \epsilon = 10^{-6}.$$ $$c_i = \lVert \mathbf{h}_i^{(o)} - \mathbf{h}_{i-1}^{(o)} \rVert_1,$$ $$\texttt{disruptive_score}_i = b_i + j_i + c_i,$$
`pseudo_change`	Sender-local pseudonym-change indicator: $$\texttt{pseudo_change}_i = \mathbb{I}[\texttt{senderPseudo}_i \neq \texttt{senderPseudo}_{i-1}].$$
`pos_error`	Position-consistency residual: $$\texttt{pos_error}_i = \left\lVert \mathbf{p}_i^{(o)} - \left(\mathbf{p}_{i-1}^{(o)} + \mathbf{v}_{i-1}^{(o)}\texttt{dt}_i\right) \right\rVert_2.$$
`pos_jitter`	Shifted local standard deviation of the position residual: $$\texttt{pos_jitter}_i = \mathrm{std}\{\texttt{pos_error}_j : j \in \mathcal{W}_{i,5}\}.$$
`zero_count_2d`	Number of zero-valued components in the eight-dimensional kinematic state: $$\texttt{zero_count_2d}_i = \sum_{\ell=1}^{8} \mathbb{I}\Big[\kappa_{i,\ell}^{(o)} = 0\Big].$$
`zero_ratio_roll_100`	Shifted proportion of all-zero kinematic reports: $$\texttt{zero_ratio_roll_100}_i = \frac{1}{\lvert\mathcal{W}_{i,100}\rvert} \sum_{j \in \mathcal{W}_{i,100}} \mathbb{I}[\texttt{zero_count_2d}_j = 8].$$
`constant_position_ratio_roll_100`	Shifted proportion of near-constant-position transitions: $$\texttt{constant_position_ratio_roll_100}_i^{(o)} = \frac{1}{\lvert\mathcal{W}_{i,100}\rvert} \sum_{j \in \mathcal{W}_{i,100}} \mathbb{I}\Big[\lVert \mathbf{p}_j^{(o)} - \mathbf{p}_{j-1}^{(o)} \rVert_2 \le \tau_p^{(o)}\Big].$$

The representation remains compact because intermediate quantities such as speed magnitude, braking intensity, jerk, heading variation, zero flags, and constant-motion flags are used only during construction. Their diagnostic contribution is preserved through aggregated descriptors such as `disruptive_score`, `zero_ratio_roll_100`, `constant_speed_ratio_roll_100`, and `constant_position_ratio_roll_100`. Short five-message rolling windows capture local instability, whereas 100-message shifted ratios capture persistent constant-speed, constant-position, or zero-pattern behavior. All rolling statistics are shifted so that the current descriptor never uses current or future label information.

All engineered descriptors are clipped to $[-10^6, 10^6]$, infinite values are converted to missing values, and remaining missing values are replaced by zero before scaling. This feature-engineering design provides a compact and physically meaningful alternative to synthetic class balancing: instead of fabricating additional minority-class trajectories, which may be statistically convenient but kinematically implausible [30,33,34], the framework preserves the original distribution while making temporal and physical attack cues explicit. The next subsection describes how these descriptors are normalized and transformed into temporal sequences for recurrent learning.

3.5. Normalization and Temporal Window Construction

After sender-level partitioning and physics-informed feature engineering, the 20 retained descriptors are normalized and reorganized into temporal tensors for recurrent learning. This step is required because the feature set combines large-scale position values, speed and acceleration components, binary indicators, rolling ratios, and residual-based descriptors. Without scaling, large-magnitude variables could dominate optimization and mask smaller but diagnostically important cues.

To mitigate this issue, the proposed pipeline applies robust feature scaling. For each observability regime o, a `RobustScaler` is fitted exclusively on the training split:

$$\text{Scaler}^{(o)} = \text{fit}\left(\mathcal{D}^{(o)}_{\text{train}}\right).$$

For a feature x_j, robust scaling is defined as

$$\widetilde{x}_j = \frac{x_j - \text{median}(x_j)}{\text{IQR}(x_j)},$$

where $\text{IQR}(x_j)$ denotes the interquartile range of feature j, estimated from the training data only. Robust scaling is appropriate here because malicious BSMs may contain extreme or physically implausible values, and median/IQR statistics reduce the influence of outliers compared with min–max normalization.

The fitted scaler is then applied without refitting to the validation and test partitions:

$$\tilde{\mathcal{D}}^{(o)}_{\text{train}} = \text{Scaler}^{(o)}\left(\mathcal{D}^{(o)}_{\text{train}}\right),\ \tilde{\mathcal{D}}^{(o)}_{\text{val}} = \text{Scaler}^{(o)}\left(\mathcal{D}^{(o)}_{\text{val}}\right) \text{ and } \tilde{\mathcal{D}}^{(o)}_{\text{test}} = \text{Scaler}^{(o)}\left(\mathcal{D}^{(o)}_{\text{test}}\right).$$

This training-only protocol prevents validation or test information from entering the preprocessing statistics. It is applied separately to ideal and realistic observability because clean and noisy BSM-derived fields have different numerical distributions.

Once normalized, the sender-specific tabular representation is transformed into temporal windows. Let

$$\widetilde{\mathbf{x}}^{(v,o)}_i \in \mathbb{R}^d$$

denote the normalized 20-dimensional feature vector associated with the i-th BSM of sender v under observability regime o, with $d = 20$. For each sender, messages are kept

in chronological order according to `sendTime`. A sliding window of length T is then constructed from the previous T normalized BSM descriptors:

$$\mathbf{X}_i^{(v,o)} = [\tilde{\mathbf{x}}_{i-T}^{(v,o)}, \tilde{\mathbf{x}}_{i-T+1}^{(v,o)}, \ldots, \tilde{\mathbf{x}}_{i-1}^{(v,o)}] \in \mathbb{R}^{T \times d}.$$

The corresponding target is the class label of the next message:

$$y_i^{(v)} = \text{class}\left(m_i^{(v,o)}\right).$$

Thus, the model receives the recent sender-local BSM history and predicts the behavioral class of the next incoming BSM. The temporal horizon is evaluated over

$$T \in \{10, 20, 30, 40, 50\},$$

which produces input tensors of shape

$$(\text{samples}, T, 20).$$

This one-step-ahead formulation avoids using the current message features to predict their own label and is well suited to replay, eventual-stop, disruptive, flooding, and Sybil-related behaviors, whose signatures may emerge progressively over several BSMs. Evaluating several values of T quantifies the benefit of additional temporal context before it increases inference cost without proportional diagnostic gain.

Temporal window construction is performed independently within each sender trajectory and within each data partition. No window is allowed to cross sender boundaries, and no window is allowed to cross train–validation–test boundaries. As established in Equation (3), a sender containing n_v messages generates $N_v(T) = \max(n_v - T, 0)$ supervised sequences.

Consequently, the total number of sequences in split $s \in \{\text{train}, \text{val}, \text{test}\}$ under observability regime o is

$$N_s^{(o)}(T) = \sum_{v \in \mathcal{V}_s} \max(n_v - T, 0). \tag{13}$$

Because the ideal and realistic regimes are derived from the same sender trajectories and use the same partitioning protocol, they produce the same number of temporal sequences for a given value of T. However, the number of sequences is not constant across temporal horizons, because the first T messages of each sender trajectory are used only as historical context. Therefore, larger values of T reduce the number of generated training, validation, and test windows. The resulting sequence counts for the evaluated temporal horizons are summarized in Table 9.

Table 9. Raw BSM partitions and temporal sequence counts for the evaluated temporal horizons.

Window Length	Input Shape	Training Sequences	Validation Sequences	Test Sequences	Total Sequences
$T = 10$	10×20	2,127,163	233,784	589,050	2,949,997
$T = 20$	20×20	1,954,044	214,537	540,892	2,709,473
$T = 30$	30×20	1,786,874	196,062	494,466	2,477,402
$T = 40$	40×20	1,625,460	178,184	449,784	2,253,428
$T = 50$	50×20	1,468,954	160,808	406,538	2,036,300

For memory efficiency, the generated tensors are stored as memory-mapped arrays, using `float32` for input sequences and `int16` for class labels. This allows the large temporal datasets to be processed without keeping all sequences in RAM, which is especially

important for $T = 40$ and $T = 50$. Training batches are loaded from disk through a sequence generator, and validation/test evaluation uses the same stored representation. These normalized and windowed tensors are the direct input to the LSTM classifier described next. Based on the sensitivity analysis in Section 4.2, $T = 40$ is retained as the main deployment-oriented configuration.

3.6. LSTM Architecture

The sequential classifier models sender-local BSM dynamics while remaining compact enough for edge-oriented compression. It receives a normalized temporal window of the previous T BSM descriptors and predicts the behavioral class of the next message. Since the final representation contains $d = 20$ descriptors, each input sample is

$$\mathbf{X} \in \mathbb{R}^{T \times 20}.$$

The output space contains $C = 20$ classes, corresponding to one normal class and 19 attack or anomalous behavior classes. The same architecture is evaluated for $T \in \{10, 20, 30, 40, 50\}$, with $T = 40$ retained as the deployment-oriented configuration.

The model follows a compact recurrent architecture composed of three stacked LSTM layers and a lightweight residual temporal-pooling branch. The first LSTM layer contains 64 hidden units and returns the full temporal sequence. The second LSTM layer contains 32 hidden units and also returns the full temporal sequence. The third LSTM layer contains 32 hidden units and returns the final temporal embedding. Dropout with rate 0.2 is applied after each recurrent layer to reduce overfitting. Formally, the recurrent backbone can be written as

$$\begin{aligned}
\mathbf{H}^{(1)} &= \text{Dropout}(\text{LSTM}_{64}(\mathbf{X})), \\
\mathbf{H}^{(2)} &= \text{Dropout}\Big(\text{LSTM}_{32}(\mathbf{H}^{(1)})\Big), \\
\mathbf{h}^{(3)} &= \text{Dropout}\Big(\text{LSTM}_{32}(\mathbf{H}^{(2)})\Big),
\end{aligned}$$

where $\mathbf{H}^{(1)}$ and $\mathbf{H}^{(2)}$ are temporal hidden representations, while $\mathbf{h}^{(3)}$ is the final recurrent embedding of the input BSM window.

To preserve coarse information from the original feature sequence, the architecture includes a residual pooling branch directly connected to the input tensor. Global Average Pooling (GAP) is applied over the temporal dimension and fused with the final recurrent embedding:

$$\mathbf{g} = \text{GAP}(\mathbf{X}) \in \mathbb{R}^{20}, \qquad \mathbf{z} = \Big[(\mathbf{h}^{(3)})^{\top}, \mathbf{g}^{\top}\Big]^{\top} \in \mathbb{R}^{52}. \tag{14}$$

The pooled vector $\mathbf{g}$ provides a low-cost summary of the normalized 20-feature trajectory, while $\mathbf{z}$ combines this summary with the 32-dimensional final LSTM embedding.

This residual fusion lets the classifier combine LSTM-level temporal abstractions with a direct summary of the original physics-informed descriptors, reducing the risk that useful low-level kinematic or behavioral information is attenuated through successive recurrent transformations.

The fused representation $\mathbf{z}$ is passed to a lightweight dense classification head. This head contains a fully connected layer with 64 hidden units and ReLU activation, followed by dropout with rate 0.2, then a second fully connected layer with 32 hidden units and ReLU activation, again followed by dropout with rate 0.2. The final output layer contains 20 units and uses the softmax activation:

$$\hat{\mathbf{y}} = \text{Softmax}(\mathbf{W}_3\phi(\mathbf{W}_2\phi(\mathbf{W}_1\mathbf{z} + \mathbf{b}_1) + \mathbf{b}_2) + \mathbf{b}_3),$$

where $\phi(\cdot)$ denotes the ReLU activation function, $\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3$ and $\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3$ are the trainable weight matrices and bias vectors of the classification head, and $\hat{\mathbf{y}} \in [0,1]^{20}$ is the predicted posterior probability distribution over the 20 behavioral classes.

The final implemented topology can therefore be summarized as:

$$\begin{aligned}(T, 20) &\rightarrow \text{LSTM}_{64} \rightarrow \text{LSTM}_{32} \rightarrow \text{LSTM}_{32}, \\ &\rightarrow \text{Concat}\Big(\mathbf{h}^{(3)}, \text{GAP}(\mathbf{X})\Big) \rightarrow \text{Dense}_{64} \rightarrow \text{Dense}_{32} \rightarrow \text{Softmax}_{20}.\end{aligned} \tag{15}$$

Only the temporal dimension changes across the sensitivity analysis; the feature dimensionality, recurrent stack, dense classification head, and compression protocol remain unchanged. This design ensures that differences observed across T values reflect the effect of temporal context length rather than changes in model capacity or input feature definition.

This architecture balances temporal expressiveness and deployment feasibility. The three recurrent layers model dependencies relevant to replay, eventual-stop, disruptive, flooding, and Sybil-related behaviors, while avoiding the parameter inflation of large ensembles, transformer-style attention, or infrastructure-dependent hybrid systems. The residual pooling branch adds limited overhead while preserving coarse sequence-level information.

The same architecture is used under both ideal and realistic BSM observability. Therefore, performance differences between the two regimes reflect the effect of clean versus noisy kinematic observability rather than an architectural confound.

To examine the effect of recurrent capacity, an architecture-width ablation was conducted under realistic observability using the retained $T = 40$ and $\tau_p^{(\text{realistic})} = 4.5$ configuration. Three stacked-LSTM width configurations were evaluated: 32/16/16, 64/32/32, and 128/64/64. Only the recurrent widths were modified. The 20-feature input, residual global-average-pooling branch, 64/32 dense classification head, dropout rate, sender-disjoint partition, optimizer, maximum number of training epochs, and TensorFlow Lite conversion procedure remained unchanged. The best checkpoint of each configuration was selected using validation accuracy. Validation weighted-F1 and macro-F1 were then used to assess class-balanced behavior, while parameter count, default-optimized TensorFlow Lite size, and controlled-runtime latency were used to evaluate the deployment trade-off. Test results were computed only after checkpoint selection and were not used to select the architecture.

The model is trained using sparse categorical cross-entropy, with the target corresponding to the class label of the next BSM following the input window:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \log\left(\hat{y}_{i,y_i}\right). \tag{16}$$

where N is the number of training sequences, y_i is the ground-truth class, and $\hat{y}_{i,y_i}$ is the predicted probability assigned to that class. Accuracy is monitored during training, while macro-F1, weighted-F1, per-class F1, confusion matrices, model size, and latency are used during final evaluation. The next subsection describes the compression and deployment variants derived from this dense LSTM baseline.

3.7. Compression and Deployment Variants

Although the proposed LSTM is compact, direct deployment of the dense Keras artifact is not necessarily optimal for OBU-level execution. Therefore, for each temporal horizon $T \in \{10, 20, 30, 40, 50\}$, the protocol evaluates four variants: Dense Keras, Compression-only TensorFlow Lite, Pruning-only stripped Keras, and Pruning + Compression TensorFlow Lite. The dense model provides the predictive reference; the compression-

only variant isolates TensorFlow Lite conversion; the pruning-only variant isolates 30% magnitude-based sparsification after fine-tuning; and the final pruning-plus-compression variant represents the deployment-oriented artifact used to select the operating value of T.

Magnitude pruning is applied using the TensorFlow Model Optimization Toolkit (TFMOT) with a constant target sparsity. After dense training, the Keras model is wrapped with pruning-aware optimization layers using `tfmot.sparsity.keras.ConstantSparsity` and `begin_step=0`, so that pruning masks are active from the beginning of fine-tuning. The pruned model is initialized from the best dense-model weights, and the transferred weights are verified numerically before pruning-aware training. The retained deployment configuration uses $s_{\text{target}} = 0.30$ and is fine-tuned for eight epochs with a reduced learning rate of 2×10^{-4}. An additional $s_{\text{target}} = 0.70$ experiment is conducted at $T = 40$ under both observability regimes to evaluate sensitivity to a substantially more aggressive pruning level. After fine-tuning, the pruning wrappers are stripped before TensorFlow Lite conversion.

All variants are evaluated on the same held-out test sequences. Keras artifacts are executed with the TensorFlow/Keras runtime, whereas TensorFlow Lite artifacts are executed with the TensorFlow Lite interpreter. The comparison reports accuracy, macro-F1, weighted-F1, per-class F1, normalized confusion matrices, serialized size, and average latency per sequence, thereby linking classification quality to edge-oriented feasibility. The proposed compression path is intentionally pragmatic: it avoids distillation, cloud assistance, federated coordination, and complex multi-stage optimization, and instead follows a reproducible dense-training → pruning-aware fine-tuning → TensorFlow Lite export workflow.

Table 10 outlines the construction and specific purpose of the four deployment variants evaluated within the proposed compression-aware protocol.

Table 10. Deployment variants evaluated in the proposed compression-aware protocol.

Variant	Construction	Purpose
Dense Keras	Original trained LSTM model	Reference model used to measure the predictive performance before deployment-oriented optimization.
Compression-only	Dense Keras model converted to default-optimized TensorFlow Lite	Isolates TensorFlow Lite conversion without pruning; floating-point I/O is retained and no full-integer, float16, or representative-dataset calibration is applied.
Pruning-only	Dense Keras model fine-tuned with a 30% constant-sparsity magnitude-pruning target, then stripped	Separates pruning effects from TensorFlow Lite conversion and quantifies the standalone impact of sparse fine-tuning.
Pruning + Compression	Stripped pruned Keras model converted to TensorFlow Lite	Final deployment-oriented variant combining pruning-aware fine-tuning with TensorFlow Lite conversion for compact edge inference.

3.8. Experimental Settings and Evaluation Metrics

This subsection summarizes the experimental configuration used to train, optimize, and evaluate the proposed IDS under both BSM observability regimes and all temporal horizons. The same protocol is applied to ideal and realistic observability: the regimes differ only in the kinematic input fields used before feature engineering, while sender-level partitioning, feature dimensionality, temporal-window candidates, LSTM architecture, training settings, compression variants, and evaluation metrics remain unchanged. Thus, performance differences between the two regimes reflect the controlled inclusion of the

dataset-provided sensor-error components, whereas differences across T values reflect the amount of sender-local temporal context.

All experiments use the fixed random seed $R = 42$. The temporal window length is treated as a sensitivity parameter, with

$$T \in \{10, 20, 30, 40, 50\}.$$

For each T, the feature dimensionality remains $d = 20$, and each model receives input tensors of shape

$$(T, 20),$$

before predicting one of $C = 20$ behavioral classes. Because sender-local temporal windowing discards the first T messages of each sender trajectory, the number of training, validation, and test sequences depends on T. The same sender-level split, feature engineering, normalization, LSTM architecture, deployment variants, and metrics are used for all temporal configurations and both observability regimes. Based on the sensitivity analysis in Section 4.2, $T = 40$ is retained as the main deployment-oriented configuration because it provides the best robustness–latency compromise under realistic BSM observability while preserving near-peak performance under ideal observability.

The dense LSTM model is trained with the Adam optimizer, an initial learning rate of 10^{-3}, sparse categorical cross-entropy as defined in Equation (16), and integer-encoded class labels. Training runs for up to 35 epochs with a batch size of 256; validation accuracy is monitored, and the best dense weights are retained for testing and compression.

After dense training, pruning-aware fine-tuning is applied to obtain the pruning-only and pruning-plus-compression variants. The pruning stage starts from the best dense weights and uses magnitude-based pruning with `tfmot.sparsity.keras.ConstantSparsity`, a target sparsity of 30%, and `begin_step=0`. The pruned model is fine-tuned for 8 epochs with a reduced learning rate of 2×10^{-4}. Pruning wrappers are then stripped to obtain a standard sparse Keras model before TensorFlow Lite conversion, allowing pruning effects to be separated from TensorFlow Lite deployment effects.

TensorFlow Lite variants are evaluated with the TensorFlow Lite interpreter rather than the original Keras runtime. This ensures that serialized model size and inference latency are measured on the exported deployment artifacts. Accordingly, dense and pruning-only variants use Keras-based inference, while compression-only and pruning-plus-compression variants use interpreter-based inference on the exported `.tflite` files.

Classification performance is evaluated using accuracy, macro-F1, weighted-F1, per-class F1-scores, and confusion matrices. Accuracy is defined as

$$\text{Accuracy} = \frac{\sum_{i=1}^{N} \mathbb{I}(\hat{y}_i = y_i)}{N},$$

where N is the number of evaluated sequences, y_i is the ground-truth label, and $\hat{y}_i$ is the predicted label.

For each class c, precision, recall, and F1-score are defined as

$$\text{Precision}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FP}_c},$$

$$\text{Recall}_c = \frac{\text{TP}_c}{\text{TP}_c + \text{FN}_c},$$

and

$$\text{F1}_c = \frac{2 \cdot \text{Precision}_c \cdot \text{Recall}_c}{\text{Precision}_c + \text{Recall}_c},$$

where TP_c, FP_c, and FN_c denote true positives, false positives, and false negatives for class c, respectively.

The macro-F1 score is computed as the unweighted average of the class-wise F1-scores:

$$\text{Macro-F1} = \frac{1}{C}\sum_{c=0}^{C-1} \text{F1}_c.$$

All macro-averaged metrics are computed over the same complete set of $C = 20$ VeReMi Extension behavior classes, indexed from 0 to 19. No class is excluded, merged, or reweighted when computing macro-F1, macro-precision, macro-recall, or the macro-average row of the class-wise analysis. This convention is used consistently for both ideal and realistic BSM observability regimes and for all deployment variants.

For interpretability of the class-wise degradation, macro-precision and macro-recall are also computed as unweighted averages over the 20 behavior classes:

$$\text{Macro-Precision} = \frac{1}{C}\sum_{c=0}^{C-1} \text{Precision}_c,$$

$$\text{Macro-Recall} = \frac{1}{C}\sum_{c=0}^{C-1} \text{Recall}_c.$$

These two auxiliary metrics are not used alone for model selection, but they help determine whether residual degradation is mainly driven by false positives or by missed detections in specific behavior classes.

Macro-F1 is central for VeReMi Extension because the class distribution is strongly imbalanced; unlike accuracy, it gives equal importance to frequent and rare classes and is therefore more informative for fine-grained multi-class diagnosis.

The weighted-F1 score is computed as

$$\text{Weighted-F1} = \sum_{c=0}^{C-1} \frac{n_c}{N}\text{F1}_c,$$

where n_c is the number of test samples belonging to class c. Weighted-F1 complements macro-F1 by incorporating empirical class frequencies while retaining class-wise F1 information.

Class-wise reports and normalized confusion matrices are generated for each variant to identify which attack families remain robust after compression and which classes are most sensitive to noisy observability or deployment-oriented optimization. This is necessary because models with similar global accuracy may behave differently on rare or safety-critical classes.

Deployment feasibility is evaluated using serialized model size and average inference latency per sequence. Model size is measured from the saved Keras artifact for dense and pruning-only variants, and from the exported `.tflite` artifact for TensorFlow Lite variants. Latency is measured in the controlled Google Colab GPU runtime used for this study, equipped with an NVIDIA T4 GPU and running TensorFlow 2.19.0 under the Python 3.12 runtime environment. The runtime corresponds to the Colab Python 3 environment version 2026.1. Although the T4 GPU is visible to TensorFlow, TensorFlow Lite evaluation is performed through `tf.lite.Interpreter` on the exported `.tflite` files without explicitly enabling an external GPU, NNAPI, or accelerator delegate. For TensorFlow Lite variants, inference is performed over the held-out test sequences in batches; the elapsed invocation time of each batch is divided by the batch size to obtain a per-sequence latency estimate, and the reported value is the mean over the evaluated test batches.

Average latency is therefore defined as:

$$\text{Latency} = \frac{1}{N}\sum_{i=1}^{N}\left(t_i^{\text{end}} - t_i^{\text{start}}\right),$$

where t_i^{start} and t_i^{end} denote the start and end timestamps associated with inference for sequence i, and N is the number of evaluated test sequences. The latency values should be interpreted as controlled runtime measurements for comparing the four model variants under the same software protocol, not as hardware-independent timing guarantees. In particular, they do not replace future validation on real OBU hardware. The complete protocol is summarized in Table 11.

Overall, this protocol serves four purposes: evaluating the physics-informed LSTM under a leakage-aware sender-level split, quantifying the gap between ideal and realistic BSM observability, measuring sensitivity to temporal window length T, and analyzing the performance–deployment trade-off induced by TensorFlow Lite conversion, pruning, and their combination. The next section reports the empirical results obtained under this protocol.

Table 11. Experimental settings used for both ideal and realistic BSM observability regimes.

Setting	Value
Random seed	42
Number of classes	20
Feature dimensionality	20 physics-informed descriptors
Temporal window lengths	$T \in \{10, 20, 30, 40, 50\}$ BSMs
Selected deployment configuration	$T = 40$ BSMs
Input tensor shape of selected model	40×20
Sequence construction	Sender-local sliding temporal windows; the number of sequences depends on T
Dense training epochs	Up to 35 epochs
Pruning fine-tuning epochs	8 epochs
Batch size	256
Execution environment	Google Colab GPU runtime, NVIDIA T4 GPU
Python runtime	Python 3.12 environment, Colab version 2026.1
TensorFlow version	2.19.0
Initial dense learning rate	10^{-3}
Pruning fine-tuning learning rate	2×10^{-4}
Pruning sparsity setting	Retained 30% target using `ConstantSparsity` with `begin_step=0`; additional 70% sensitivity experiment at $T = 40$ under both observability regimes
TensorFlow Lite conversion	Default optimization with selected TensorFlow operations; floating-point I/O; no representative dataset, float16, or full-integer quantization
Evaluated variants	Dense Keras, Compression-only, Pruning-only, Pruning + Compression
Classification metrics	Accuracy, macro-F1, weighted-F1, per-class F1, classification report, confusion matrix
Deployment metrics	Serialized model size and average inference latency per sequence
Latency interpretation	Controlled runtime comparison of exported artifacts; not a real-OBU benchmark

3.9. Generative AI Usage

During the preparation of this work, the authors used Google Gemini and OpenAI ChatGPT to assist in writing and debugging Python scripts for dataset preprocessing and processing, and to refine the English language of the manuscript. The authors explicitly state that all data analysis, interpretation of results, and formulation of conclusions are entirely the original work of the human authors. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the final content of the publication.

4. Results

This section reports the empirical results obtained with the proposed temporal-window-aware physics-informed edge IDS under the two BSM observability regimes defined in Section 3. The analysis proceeds through eleven complementary stages. First, we validate the leakage-aware sender-level split and temporal sequence construction. Second, we evaluate temporal-window sensitivity. Third, after retaining $T = 40$, we compare the four deployment variants under ideal and realistic BSM observability. Fourth, we analyze the performance–deployment trade-off in terms of macro-F1, latency, and serialized footprint. Fifth, we evaluate recurrent-width sensitivity. Sixth, we compare controlled temporal-model baselines. Seventh, we study pruning-sparsity sensitivity. Eighth, we report large-scale and hardware-constrained inference benchmarks. Ninth, we perform an architecture-preserving feature-family ablation. Tenth, we examine combined feature-group ablations. Eleventh, we analyze final class-wise behavior under ideal and realistic observability.

For conciseness, the labels "ideal" and "realistic" are retained throughout this section. As formally defined in Section 3.1, both conditions are simulation-based, and "realistic" denotes the noise-inclusive VeReMi Extension condition rather than real-world vehicle or wireless-link validation. The four compared variants should likewise be interpreted as implementation and export variants whose footprint and runtime measurements quantify edge-deployment potential under the tested platforms.

4.1. Dataset Splits and Sequence Construction

Table 9 confirms the sender-disjoint partition and the resulting sequence counts under both observability regimes. Because the first T messages of each sender provide historical context, increasing T reduces the number of supervised sequences according to Equation (3). The two observability regimes retain identical senders, timestamps, labels, and sequence counts, and differ only in their observable kinematic fields.

The sequence-label distributions preserve the strong class imbalance of the original VeReMi Extension benchmark. For the selected $T = 40$ configuration, class 0 remains dominant with 892,919 training sequences, corresponding to 54.93% of the training set. The remaining 19 classes are substantially less frequent, with several attack classes representing only about 1–1.4% of the training sequences, while classes 13, 14, 15, 16, 18, and 19 are more represented but still minority classes. This confirms that the proposed pipeline does not rely on synthetic oversampling or artificial class balancing.

The distributions in Table 12 show that the held-out $T = 40$ test set remains strongly imbalanced: class 0 represents 55.63% of the test sequences, several attack classes account for only about 1–1.5%, and classes 13, 14, 15, 16, 18, and 19 have higher but still minority support. This confirms that the selected test configuration remains demanding and justifies reporting macro-F1 and class-wise F1 in addition to accuracy.

Overall, the preprocessing results confirm three properties of the protocol. First, sender-level partitioning prevents trajectory leakage between training, validation, and test

sets. Second, the same sequence construction is applied under both observability regimes and across all temporal horizons, making the ideal–realistic comparison and the temporal-window sensitivity analysis controlled. Third, the original class imbalance is preserved, so subsequent results reflect the model's ability to learn discriminative physics-informed representations without synthetic class rebalancing.

Table 12. Sequence-label distribution after sender-level splitting and temporal window construction for the selected $T = 40$ configuration.

Class	Training		Validation		Test	
	Count	Percent	Count	Percent	Count	Percent
0	892,919	54.9333	100,721	56.5264	250,231	55.6336
1	20,295	1.2486	2958	1.6601	5526	1.2286
2	21,248	1.3072	2424	1.3604	5079	1.1292
3	19,805	1.2184	3128	1.7555	6070	1.3495
4	19,295	1.1870	1932	1.0843	6512	1.4478
5	19,747	1.2149	1781	0.9995	5546	1.2330
6	21,621	1.3301	2108	1.1830	5788	1.2868
7	20,571	1.2655	1439	0.8076	5583	1.2413
8	19,778	1.2168	2195	1.2319	5811	1.2920
9	19,824	1.2196	2206	1.2380	6317	1.4045
10	20,530	1.2630	2222	1.2470	5635	1.2528
11	22,079	1.3583	1902	1.0674	5393	1.1990
12	20,664	1.2713	2461	1.3812	5025	1.1172
13	86,211	5.3038	5068	2.8443	24,565	5.4615
14	74,662	4.5933	10,814	6.0690	25,913	5.7612
15	84,445	5.1951	9448	5.3024	20,011	4.4490
16	120,927	7.4396	12,636	7.0915	26,450	5.8806
17	20,478	1.2598	1544	0.8665	7424	1.6506
18	53,351	3.2822	6016	3.3763	12,341	2.7438
19	47,010	2.8921	5181	2.9077	14,564	3.2380
Total	**1,625,460**	**100.0000**	**178,184**	**100.0000**	**449,784**	**100.0000**

4.2. Temporal Window-Length Sensitivity

Before analyzing the four deployment variants in detail, we evaluate the effect of temporal window length on the final deployment-oriented model. This analysis is important because the choice of T directly controls the amount of sender-local BSM history available to the LSTM. A short window may be insufficient to capture progressive or persistent misbehavior patterns, whereas an excessively long window may increase inference latency and reduce the number of available supervised sequences without providing proportional diagnostic gains.

To quantify the observability effect at each temporal horizon, we define the macro-F1 observability gap as

$$\Delta_{\text{obs}}(T) = F_{1,\text{macro}}^{(\text{ideal})}(T) - F_{1,\text{macro}}^{(\text{realistic})}(T). \tag{17}$$

For $T \in \{10, 20, 30, 40, 50\}$, the measured gaps, expressed in percentage points, are

$$[\Delta_{\text{obs}}(10), \Delta_{\text{obs}}(20), \Delta_{\text{obs}}(30), \Delta_{\text{obs}}(40), \Delta_{\text{obs}}(50)] = [0.48, 0.48, 0.30, 0.42, 0.69].$$

The input-level exposure to the dataset-provided sensor error can also be expressed explicitly. Let

$$\mathbf{n}_i = [\texttt{posx_n}_i, \texttt{posy_n}_i, \texttt{spdx_n}_i, \texttt{spdy_n}_i, \texttt{aclx_n}_i, \texttt{acly_n}_i, \texttt{hedx_n}_i, \texttt{hedy_n}_i]^\top$$

denote the eight-dimensional sensor-error vector associated with message i, and let

$$\mathbf{N}_{i,T} = [\mathbf{n}_{i-T}, \ldots, \mathbf{n}_{i-1}]^\top$$

denote the error tensor exposed over a temporal window of length T. Under the simplifying assumptions of temporally stationary, zero-mean, and uncorrelated sensor errors with covariance matrix $\mathbf{\Sigma}_n$,

$$\mathbb{E}\left[\|\mathbf{N}_{i,T}\|_F^2\right] = T\,\mathrm{tr}(\mathbf{\Sigma}_n). \tag{18}$$

Here, $\mathbb{E}[\cdot]$ denotes mathematical expectation, $\|\cdot\|_F$ is the Frobenius norm, and $\mathrm{tr}(\cdot)$ denotes the matrix trace.

Thus, increasing the window from $T = 40$ to $T = 50$ increases the expected total input-level noise energy by 25%, although it does not increase the expected noise energy per message. Because robust scaling, nonlinear feature engineering, and LSTM processing transform these perturbations, Equation (18) should be interpreted as a quantitative measure of input exposure rather than as a direct causal model of classification error.

The results in Table 13 demonstrate the relationship between temporal window length, observability, and inference latency. Under ideal observability, extending the window from $T = 10$ to $T = 50$ yields a consistent improvement in diagnostic capacity, with macro-F1 rising from 98.45% to a peak of 99.17%. Under realistic BSM observability, the model exhibits a similar progression up to $T = 40$, where it reaches its maximum macro-F1 of 98.67% and an accuracy of 99.38%.

Table 13. Temporal window-length sensitivity of the final pruning-plus-compression TensorFlow Lite model under ideal and realistic BSM observability (Sparsity = 30%).

Window	Ideal BSM Observability			Realistic BSM Observability		
T	Accuracy (%)	Macro-F1 (%)	Latency (ms)	Accuracy (%)	Macro-F1 (%)	Latency (ms)
10	99.33	98.45	0.0419	99.09	97.97	0.0371
20	99.51	98.97	0.0821	99.30	98.49	0.0752
30	99.52	98.96	0.0959	99.37	98.66	0.1079
40	99.60	99.09	0.1398	**99.38**	**98.67**	0.1283
50	**99.61**	**99.17**	0.1635	99.33	98.48	0.1680

Raw predictive metrics must be balanced against deployment constraints and physical motion semantics. Inference latency increases with the temporal dimension in both regimes, rising under realistic observability from 0.0371 ms at $T = 10$ to 0.1680 ms at $T = 50$. The $T = 40$ configuration maintains a low controlled-runtime inference latency of 0.1283 ms while providing a 40-message temporal receptive field capable of capturing progressive long-horizon attacks, such as Eventual Stop and subtle trajectory offsets, that may be less visible in shorter observation frames such as $T = 20$ or $T = 30$.

Extending the window from $T = 40$ to $T = 50$ produces different effects under the two observability regimes. Under ideal observability, macro-F1 increases by 0.08 percentage points, from 99.09% to 99.17%. Under realistic observability, it decreases by 0.19 percentage points, from 98.67% to 98.48%, while the observability gap increases from 0.42 to 0.69 percentage points. At the same time, the number of test sequences decreases from 449,784 to 406,538, corresponding to a reduction of approximately 9.61%, and controlled-runtime latency increases from 0.1283 ms to 0.1680 ms, corresponding to an increase of approximately 30.94%. These observations are consistent with a combination of greater cumulative exposure to sensor-error components, fewer supervised sequences, redundant

long-range context, and increased computational cost. They do not establish sensor noise as the sole causal mechanism. Consequently, $T = 40$ is retained as the most favorable empirical robustness–latency compromise under realistic observability rather than as a universal optimum.

4.3. Aggregate Results Under Ideal and Realistic BSM Observability

After selecting $T = 40$ as the main deployment-oriented temporal configuration, the four model variants are evaluated under both ideal and realistic BSM observability. The ideal regime provides an upper-bound reference based on clean simulator-side kinematic fields, whereas the realistic regime is the main deployment-oriented reference because it uses noisy BSM-derived observables. In both cases, the same sender-level split, feature-engineering logic, temporal window length, LSTM architecture, compression protocol, and evaluation metrics are used, so the comparison isolates the effects of observability and deployment optimization.

Table 14 summarizes the aggregate performance of the four variants. Under ideal BSM observability, the dense Keras baseline achieves the highest score, with 99.63% accuracy and 99.19% macro-F1. The final pruning-plus-compression TensorFlow Lite model remains highly competitive, preserving 99.60% accuracy and 99.09% macro-F1. Under realistic BSM observability, the dense baseline reaches 99.48% accuracy and 98.85% macro-F1, while the final pruning-plus-compression model maintains 99.38% accuracy and 98.67% macro-F1. Relative to the realistic dense baseline, the final compressed model loses only 0.10 percentage points in accuracy and 0.18 percentage points in macro-F1, confirming that pruning-aware fine-tuning and TensorFlow Lite conversion introduce minimal aggregate degradation.

Table 14. Aggregate performance of the four deployment variants under ideal and realistic BSM observability for the selected $T = 40$ configuration.

Model Variant	Ideal BSM Observability			Realistic BSM Observability		
	Acc. (%)	Macro-F1 (%)	Weighted-F1 (%)	Acc. (%)	Macro-F1 (%)	Weighted-F1 (%)
Dense Keras	99.63	99.19	99.63	99.48	98.85	99.46
Compression-only	99.60	99.14	99.59	99.47	98.83	99.46
Pruning-only	99.59	99.09	99.58	99.42	98.75	99.40
Pruning + Compression	99.60	99.09	99.58	99.38	98.67	99.36

The observability gap also remains limited for the final pruning-plus-compression artifact. Moving from ideal to realistic BSM observability reduces accuracy from 99.60% to 99.38%, and macro-F1 from 99.09% to 98.67%, corresponding to gaps of 0.22 and 0.42 percentage points, respectively. This confirms that noisy BSM-derived features slightly affect performance but do not compromise the fine-grained diagnostic capability of the proposed IDS. Overall, the dense model provides the strongest upper-bound reference, whereas the pruning-plus-compression TensorFlow Lite model provides the retained deployment-oriented operating point. The realistic $T = 40$ compressed model is therefore used as the main reference in the deployment and class-wise analyses that follow.

4.4. Compression, Model Size, and Latency Results

The previous subsection showed that pruning and TensorFlow Lite conversion preserve aggregate diagnostic performance under both observability regimes. This subsection analyzes whether this accuracy retention is accompanied by a practical reduction in model

footprint and inference latency. The comparison is performed for the selected $T = 40$ configuration and includes the same four variants: Dense Keras, Compression-only TensorFlow Lite, Pruning-only stripped Keras, and Pruning + Compression TensorFlow Lite.

The results show that TensorFlow Lite conversion is the primary contributor to footprint and latency reduction. Under ideal observability, the dense Keras model occupies 636.06 KB and requires 28.9998 ms per sequence, whereas the final Pruning + Compression TensorFlow Lite model occupies 88.38 KB and reaches 0.1398 ms per sequence. Under realistic observability, the dense Keras model occupies 636.06 KB and requires 28.6022 ms per sequence, while the final Pruning + Compression TensorFlow Lite model occupies 88.38 KB and executes in 0.1283 ms per sequence. Thus, the deployment artifact reduces the serialized model size by approximately 7.20×, corresponding to an 86.11% footprint reduction, and accelerates inference by 207.46× under ideal observability and 222.96× under realistic observability, while preserving strong macro-F1 performance.

These latency values are intended to compare the relative deployment behavior of the four exported variants under a fixed software protocol. They should not be interpreted as direct measurements on a production OBU. In particular, the TensorFlow Lite values characterize the exported interpreter-based artifacts in the experimental runtime, whereas real vehicular hardware may introduce different CPU architectures, memory hierarchies, operating-system scheduling effects, accelerator availability, and I/O overheads. The OBU claim is therefore a deployability indication based on compact footprint and low controlled-runtime inference latency, not a substitute for hardware-in-the-loop validation.

Figure 2 visualizes the trade-off between macro-F1, inference latency, and serialized footprint for the four variants. It shows that the dense Keras baseline remains the strongest predictive reference but occupies the high-latency, larger-footprint region, whereas the TensorFlow Lite variants shift sharply toward compact and low-latency inference with only minimal macro-F1 degradation.

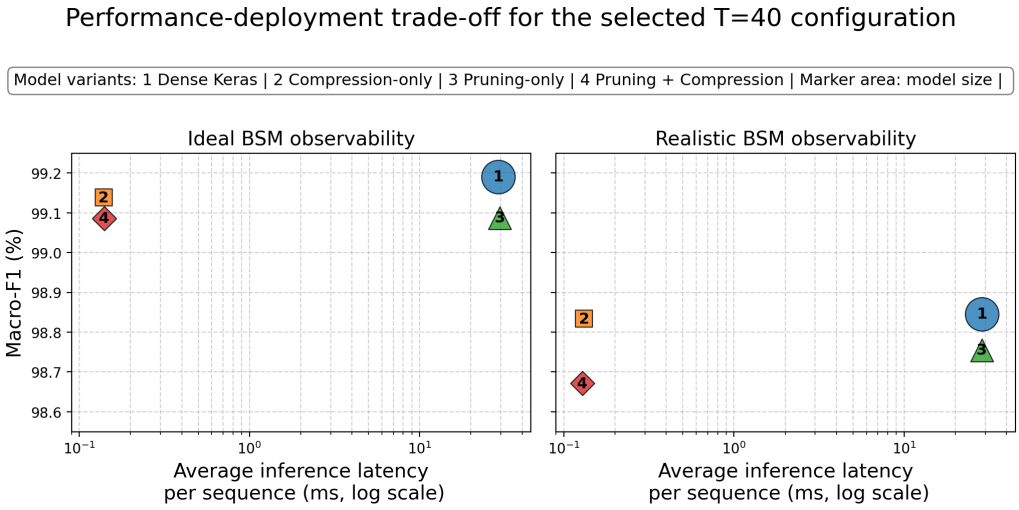


Figure 2. Performance–deployment trade-off of the four model variants under the selected $T = 40$ configuration. The horizontal axis reports average inference latency per sequence on a logarithmic scale, the vertical axis reports macro-F1, and marker area represents serialized model footprint. Numbered markers identify the four deployment variants: (1) Dense Keras, (2) Compression-only, (3) Pruning-only, and (4) Pruning + Compression.

The final Pruning + Compression artifact remains below 90 KB in both observability regimes, reaches 0.1283 ms under the realistic controlled-runtime regime, and preserves high macro-F1. These results provide evidence of a compact and low-latency exported

artifact, while actual local edge-oriented inference still requires validation on representative OBU or roadside-edge hardware.

4.5. LSTM Architecture-Width Ablation at $T = 40$

The architecture-width ablation compares three recurrent capacities under the same realistic-observability configuration.

Table 15 reports the best validation checkpoint of each architecture together with the deployment characteristics of its default-optimized TensorFlow Lite artifact.

This architecture-width study is a dedicated controlled retraining and export experiment. Its validation metrics, confirmatory test metrics, serialized sizes, and latency measurements are therefore used for within-ablation comparison across the three widths. They do not replace the canonical end-to-end test results of the retained 64/32/32 dense, compression-only, pruning-only, and pruning-plus-compression pipeline reported in Table 14 and Section 4.4.

Table 15. Architecture-width ablation under realistic observability at $T = 40$ and $\tau_p^{(\text{realistic})} = 4.5$ m. The † symbol identifies the retained performance–deployment configuration.

LSTM Widths	Best Epoch	Val. Acc. (%)	Val. Weighted-F1 (%)	Val. Macro-F1 (%)	Parameters	TFLite Size (KB)	Latency (ms)
32/16/16	34	99.307	99.295	98.640	17,140	50.81	0.0752
64/32/32 †	13	99.355	99.337	98.787	48,628	88.31	0.1444
128/64/64	9	99.370	99.357	98.789	166,900	217.31	0.3347

The 32/16/16 configuration provides the smallest and fastest artifact, but its validation macro-F1 is 0.1470 percentage points lower than that of 64/32/32. Increasing the recurrent widths from 64/32/32 to 128/64/64 produces only marginal validation improvements of 0.0152 percentage points in accuracy, 0.0207 points in weighted-F1, and 0.0019 points in macro-F1. In contrast, the larger architecture requires 3.43× more parameters, a 2.46× larger TensorFlow Lite artifact, and 2.32× higher controlled-runtime latency.

The corresponding default-optimized TensorFlow Lite test accuracy, weighted-F1, and macro-F1 values are respectively 99.391%, 99.381%, and 98.649% for 32/16/16; 99.416%, 99.400%, and 98.787% for 64/32/32; and 99.467%, 99.453%, and 98.818% for 128/64/64. These held-out results confirm the limited predictive benefit of the largest architecture but were not used for architecture selection.

The 64/32/32 configuration therefore captures essentially all of the class-balanced validation performance of 128/64/64, while remaining below 90 KB and 0.15 ms in the controlled TensorFlow Lite evaluation. It is retained as the knee point of the performance–deployment trade-off rather than as a universally optimal recurrent architecture.

4.6. Controlled Temporal-Model Baselines at $T = 40$

To separate the contribution of the recurrent-cell family from that of the proposed 20-feature representation, five temporal-learning configurations were compared under the same realistic-observability protocol. All configurations used the same sender-disjoint train/validation/test partitions, random seed, $T = 40$, training-only RobustScaler, $\tau_v = 0.025$ m/s, $\tau_p^{(\text{realistic})} = 4.5$ m, feature vector, batch size, optimizer, maximum training budget, 64/32 dense classification head, dropout rate, checkpoint-selection rule, and TensorFlow Lite export procedure. The LSTM, GRU, and SimpleRNN candidates used the same 64/32/32 recurrent widths and residual global-average-pooling branch; only the recurrent-cell family was changed. Two non-recurrent controls were also included: GAP-

MLP averages the $T = 40$ input vectors before the common dense head, whereas Last-step MLP uses only the most recent descriptor vector.

The best checkpoint of each candidate was selected exclusively using validation accuracy. Validation weighted-F1 and macro-F1 were then computed to assess class-balanced behavior. The held-out test set was evaluated only after checkpoint selection and was not used for model choice. Table 16 reports the validation results and the deployment characteristics of the corresponding default-optimized TensorFlow Lite artifacts.

The LSTM reference row reproduces the dedicated 64/32/32 checkpoint used in the architecture-width study, while the other rows are trained and exported under the same controlled comparison protocol; these values are not substitutes for the four-variant main-pipeline results in Table 14.

Table 16. Controlled temporal-model comparison under realistic observability at $T = 40$, $\tau_v = 0.025$ m/s, and $\tau_p^{(\text{realistic})} = 4.5$ m. Validation metrics are computed from the selected dense checkpoint; size and latency refer to its default-optimized TensorFlow Lite export. The † symbol identifies the retained model.

Temporal Model	Widths	Val. Acc. (%)	Val. Weighted-F1 (%)	Val. Macro-F1 (%)	Parameters	TFLite Size (KB)	Latency (ms)	Best Epoch
LSTM †	64/32/32	99.355	99.337	98.787	48,628	88.30	0.1377	13
GRU	64/32/32	99.242	99.219	98.511	38,388	76.74	0.1204	22
SimpleRNN	64/32/32	98.115	97.440	93.974	16,756	43.46	0.0626	34
GAP-MLP	–	98.477	98.260	96.093	4084	9.61	0.0012	35
Last-step MLP	–	97.151	96.465	90.275	4084	10.09	0.0023	32

The LSTM provides the highest validation accuracy, weighted-F1, and macro-F1 among the controlled candidates. Relative to LSTM, GRU reduces the parameter count by 21.06%, the TensorFlow Lite footprint by 13.09%, and the controlled-runtime latency by 12.59%, but decreases validation accuracy, weighted-F1, and macro-F1 by 0.113, 0.118, and 0.276 percentage points, respectively. SimpleRNN is substantially smaller and faster, but its validation macro-F1 is 4.813 percentage points lower than that of LSTM. The non-recurrent GAP-MLP and Last-step MLP controls reduce validation macro-F1 by 2.694 and 8.512 percentage points, respectively, showing that the engineered descriptors remain discriminative but that learned temporal recurrence provides additional class-balanced discrimination.

Confirmatory held-out evaluation of the default-optimized TensorFlow Lite artifacts yielded the following accuracy, weighted-F1, and macro-F1 values, respectively: 99.416%, 99.400%, and 98.787% for LSTM; 99.303%, 99.280%, and 98.453% for GRU; 98.337%, 97.773%, and 94.041% for SimpleRNN; 98.169%, 97.852%, and 94.743% for GAP-MLP; and 89.409%, 88.026%, and 75.420% for Last-step MLP. These test results were not used for model selection.

The controlled comparison therefore supports the retained LSTM as the best measured accuracy–class-balance choice under the common protocol. GRU remains a viable smaller alternative when footprint and latency are prioritized over the observed macro-F1 difference. The comparison also separates the role of the representation from that of the temporal learner: all five candidates receive the same physics-informed descriptors, while the performance differences arise from how their temporal context is aggregated. Because previously published systems often use different class mappings, observability assumptions, feature pipelines, and data partitions, these controlled baselines are not presented as exact reproductions of systems whose complete end-to-end protocols are not available in a directly reusable, protocol-equivalent form.

4.7. Sparsity-Level Sensitivity at $T = 40$

To assess whether the retained pruning configuration represents a reasonable deployment compromise, an additional sparsity experiment was conducted at $T = 40$ using target sparsities of 30% and 70%. The same sender-level split, 20-feature representation, LSTM architecture, dense-weight initialization, pruning-aware fine-tuning protocol, and TensorFlow Lite conversion procedure were used for both targets and observability regimes. The comparison therefore isolates the effect of the pruning target. The results for the final Pruning + Compression TensorFlow Lite artifacts are reported in Table 17.

Table 17. Sensitivity of the final Pruning + Compression TensorFlow Lite model to the pruning target at $T = 40$.

Observability	Target Sparsity	Accuracy (%)	Macro-F1 (%)	Weighted-F1 (%)	TFLite Size (KB)
Ideal	30%	99.60	99.09	99.58	88.38
Ideal	70%	99.52	98.89	99.51	88.38
Realistic	30%	99.38	98.67	99.36	88.40
Realistic	70%	99.36	98.56	99.34	88.40

Increasing the pruning target from 30% to 70% produces a small but consistent degradation under both observability regimes. Under ideal observability, accuracy decreases by 0.07 percentage points and macro-F1 decreases by 0.20 percentage points. Under realistic observability, the corresponding decreases are 0.02 and 0.11 percentage points. The realistic 70%-sparsity stripped Keras artifact was additionally inspected after training and contained exactly 33,600 zero-valued matrix weights out of 48,000, confirming an effective matrix-weight sparsity of 70.00%.

The serialized TensorFlow Lite footprint remains essentially unchanged when sparsity increases from 30% to 70%. This occurs because the default TensorFlow Lite conversion used in this work does not encode the additional unstructured zeros into a smaller sparse deployment representation. Consequently, the 70% target introduces a larger predictive degradation without providing an additional serialized-footprint advantage under the current export pipeline. The 30% target is therefore retained as the conservative pruning configuration because it preserves higher class-balanced performance while producing the same compact TensorFlow Lite footprint. This choice should not be interpreted as a universally optimal sparsity level; structured sparsity and sparsity-aware runtime acceleration remain subjects for future investigation.

4.8. Large-Scale and Single-Thread TensorFlow Lite Runtime Benchmarks

To complement the controlled-runtime TensorFlow Lite measurements, we conducted two dedicated CPU inference benchmarks using test sequences from the simulation-based realistic BSM regime.

First, to evaluate high-throughput processing capacity, both the dense and the pruned-and-compressed TensorFlow Lite models were evaluated on 150,000 held-out test sequences using the retained temporal window size $T = 40$. The benchmark was executed using the Python interpreter on a multi-core Intel Xeon CPU running at 2.20 GHz in a Google Colab virtualized environment with GPU execution disabled. Table 18 summarizes the results. The pruned TensorFlow Lite model achieves 99.14% accuracy and 98.18% macro-F1 while maintaining an average inference latency of 3.544 ms and a throughput of 282 sequences per second. The performance gap relative to the dense model remains very limited (0.12 percentage points in accuracy and 0.23 percentage points in macro-F1).

Table 18. Large-scale CPU inference benchmark results using the simulation-based realistic (noise-inclusive) BSM regime at the retained $T = 40$ configuration (150,000 sequences).

Model	Accuracy (%)	Macro-F1 (%)	Avg. Latency (ms)	Throughput (seq/s)
Dense (Keras)	99.26	98.41	3.398	294.3
Pruned + TFLite	99.14	98.18	3.544	282.2

Second, the final pruned-and-compressed TensorFlow Lite artifact was profiled using the official C++ TensorFlow Lite Benchmark Tool (`benchmark_model_plus_flex`). This complementary experiment removes Python interpreter and batching effects from the measured inference path. The benchmark was configured with one CPU thread (`num_threads = 1`) and unbatched sequential inputs. It should therefore be interpreted as a controlled single-thread runtime characterization, not as a measurement performed on a representative OBU or RSU.

Under these conditions, the benchmarking tool reports an estimated runtime memory footprint of 90.49 KB, close to the 88.38 KB serialized disk size. The initial warm-up time is 16.79 ms, and the average unbatched per-sequence inference latency is 1.356 ms. Since BSMs are commonly generated at approximately 10 Hz, the measured inference latency corresponds to about 1.36% of a 100 ms message interval. This result supports the artifact's edge-deployment potential under the evaluated C++ runtime and CPU configuration. It is not a production OBU guarantee because end-to-end deployment additionally involves message reception, parsing, feature construction, scheduling, memory hierarchy, communication-stack behavior, and the target processor architecture.

4.9. Feature-Family Ablation Under Realistic BSM Observability

To quantify the contribution of the proposed physics-informed descriptor families, we conduct an architecture-preserving training-time feature-family ablation under the retained realistic $T = 40$ configuration. For each scenario, the selected feature family is masked to the neutral value 0 in the scaled feature space, after fitting the robust scaler on the training split only. The same masking is applied consistently to the training, validation, and test partitions. Each ablated configuration is trained from scratch and evaluated across the full deployment chain: Dense Keras, Compression-only TensorFlow Lite, Pruning-only Keras, and Pruning + Compression TensorFlow Lite. This protocol keeps the sender-level partitioning, input dimensionality, temporal window size, LSTM architecture, pruning target, and conversion pipeline fixed, ensuring that performance differences primarily reflect the informational contribution of the masked descriptors.

Table 19 reports macro-F1 for all four deployment variants. The hierarchy of feature importance remains stable from the dense model to the final Pruning + Compression TensorFlow Lite artifact. With the complete 20-feature representation, macro-F1 remains high across the deployment chain, decreasing only from 98.85% for Dense Keras to 98.67% for Pruning + Compression. This confirms that the deployment pipeline does not substantially alter the class-balanced behavior of the model and that the feature-importance hierarchy is not an artifact of a single runtime variant.

The most severe degradation occurs when all twelve engineered non-raw descriptors are masked, retaining only the eight raw kinematic channels. In this configuration, macro-F1 drops to 67.62% for the dense model and 67.49% for the final exported artifact, corresponding to a 31.18-point loss relative to the full exported-artifact baseline. This result is important because raw position, speed, acceleration, and heading describe the instantaneous state of a BSM, but they do not directly encode whether the state is temporally consistent, whether the message has been replayed, whether the sender identity evolves abnormally, or whether the reported motion agrees with the previous sender-local trajectory.

The large drop therefore shows that the proposed IDS does not merely learn from raw kinematics; it relies on temporal, identity, replay, and physics-consistency evidence that makes multi-class misbehavior separable under realistic BSM observability.

Table 19. Training-time feature-family ablation under realistic BSM observability at the retained $T = 40$ configuration. The table reports macro-F1 for the four deployment variants. The last performance column reports the final exported Pruning + Compression TensorFlow Lite artifact, and Δ is measured in percentage points relative to the full 20-feature deployed baseline.

Ablation Scenario	Masked Features	Dense Macro-F1 (%)	Compressed Macro-F1 (%)	Pruned Macro-F1 (%)	Pruned + Comp. Macro-F1 (%)/Δ	Impact
Full 20-feature representation	0	98.85	98.83	98.75	98.67/0.00	Reference
Raw kinematics only	12	67.62	67.68	67.52	67.49/−**31.18**	**Critical**
Without pseudonym-dynamics descriptor	1	89.25	89.06	89.72	89.43/−**9.24**	**Very high**
Without position-consistency descriptors	2	96.79	96.87	97.01	97.05/−**1.62**	**High**
Without timing/replay cues	2	97.81	97.66	97.67	97.68/−0.99	Medium
Without zero/constant-position patterns	3	98.18	98.15	97.97	97.96/−0.71	Medium
Without disruptive-motion descriptor	1	98.61	98.56	98.44	98.43/−0.24	Low
Without stop/speed persistence	3	98.57	98.57	98.59	98.61/−0.06	Low

Among individual descriptor families, **pseudonym dynamics** has the strongest impact. Masking the single `pseudo_change` feature reduces deployed macro-F1 to 89.43%, a 9.24-point loss. This high sensitivity is physically and semantically consistent with the VeReMi Extension taxonomy. Several classes involve Sybil or identity-dependent behavior, including Data Replay Sybil, Traffic Congestion Sybil, DoS Random Sybil, and DoS Disruptive Sybil. In these cases, maliciousness is not always expressed only through impossible motion; it also appears through abnormal identity evolution, forged pseudonym usage, or multiple identities producing coordinated kinematic patterns. Once pseudonym dynamics are masked, the model must infer these behaviors only from timing and motion cues, which is substantially harder under noisy realistic BSM observability. The strong and stable degradation across Dense, Compression-only, Pruning-only, and Pruning + Compression variants therefore indicates that identity evolution is a central evidence channel for fine-grained Sybil-related diagnosis.

Position-consistency descriptors form the second most influential individual feature group. Removing `pos_error` and `pos_jitter` reduces deployed macro-F1 to 97.05%, corresponding to a 1.62-point loss. These descriptors are physically important because they compare the observed current position with the position predicted from the previous position and velocity. They therefore encode displacement–speed coherence, which is directly relevant to Constant Position Offset, Random Position, Random Position Offset, Disruptive Attack, and replay-like behaviors. Under realistic observability, position noise can partially blur small spatial offsets; the explicit residual features help the LSTM distinguish benign localization uncertainty from systematic position falsification. Their ablation is also consistent with the class-wise analysis, where Class 2, Constant Position Offset, remains the most difficult realistic-observability class.

Timing/replay cues have a medium impact, with a deployed macro-F1 loss of 0.99 points. This is expected because replay and flooding behaviors are not defined only by the content of a BSM, but also by when messages arrive and whether identifiers are reused. The `dt` and `msg_repeated` descriptors therefore provide communication-level evidence that complements the kinematic trajectory. When these cues are masked, the model can still exploit motion consistency and identity behavior, which explains why the drop is smaller than for pseudonym dynamics, but the degradation remains non-negligible because replay and DoS-related classes lose a direct evidence channel.

Zero-pattern and constant-position descriptors also produce a medium degradation, with a 0.71-point deployed macro-F1 loss. These features capture degenerate or persistent motion reports, including repeated zero-valued kinematic states and nearly fixed positions. They are especially relevant to Constant Position, stop-like anomalies, malformed motion, and some flooding-related behaviors. Their impact is smaller than that of position consistency because part of their information can be reconstructed indirectly from raw position, speed, and long-horizon temporal context. Nevertheless, their ablation confirms that explicit pattern descriptors help stabilize rare and physically degenerate classes.

The **disruptive-motion** descriptor and the **stop/speed-persistence** family have lower aggregate impact, with deployed macro-F1 losses of 0.24 and 0.06 points, respectively. This should not be interpreted as evidence that these descriptors are useless. Rather, it indicates that their information overlaps with remaining kinematic, position-consistency, and temporal descriptors after retraining. For example, abrupt braking, jerk-like acceleration variation, heading change, speed change, and eventual-stop behavior can be partially inferred from the raw velocity and acceleration histories already available inside the $T = 40$ temporal window. These families therefore act more as redundant stabilizers than as unique evidence channels at the aggregate macro-F1 level.

Overall, the ablation reveals three levels of feature importance. The first level is *critical*: raw kinematics alone are insufficient, producing a 31.18-point deployed macro-F1 loss. The second level is *high-impact*: pseudonym dynamics and position consistency provide distinctive identity and physics-consistency evidence that cannot be fully recovered from the remaining descriptors. The third level is *complementary*: timing/replay, zero/constant-position, disruptive-motion, and stop/speed-persistence descriptors improve robustness or stabilize specific attack families, but their information is partially redundant with other temporal and kinematic cues. This hierarchy supports the central design of the proposed IDS: accurate 20-class misbehavior detection under realistic BSM observability requires combining raw motion fields with compact temporal, identity, replay, and physics-consistency descriptors rather than relying on raw kinematics or black-box temporal learning alone.

Combined Feature-Group Analysis

To complement the individual-family ablations, three targeted combined ablations were conducted under the same realistic-observability configuration at (T = 40). The evaluated pairs were: (i) pseudonym dynamics with position consistency; (ii) pseudonym dynamics with timing and replay cues; and (iii) zero-pattern and constant-position descriptors with stop and speed-persistence descriptors. For every scenario, all selected descriptors were masked jointly to the neutral value (0) in the scaled feature space for the training, validation, and test partitions. The sender-disjoint split, 20-dimensional input, (64/32/32) LSTM architecture, pruning target, and four-variant deployment chain were kept unchanged.

Table 20 shows that jointly masking pseudonym dynamics and position consistency causes the largest degradation: the deployed macro-F1 decreases from 98.67% to 86.90%, a loss of 11.77 percentage points. Jointly masking pseudonym dynamics and timing/replay cues produces a similarly large loss of 11.13 points, reducing deployed macro-F1 to 87.54%. These results reinforce the central role of pseudonym evolution in distinguishing Sybil-related and identity-dependent classes, while also showing that position consistency and communication-level timing evidence provide complementary information.

The third combined ablation is less severe but remains non-negligible. Jointly masking zero-pattern and constant-position descriptors together with stop and speed-persistence descriptors reduces deployed macro-F1 to 97.65%, a 1.02-point loss. This decrease is larger than either constituent individual-family loss, indicating that the two groups jointly

stabilize degenerate and persistent-motion behaviours even though each group has a relatively small aggregate effect when removed alone.

Table 20. Combined feature-group ablation under realistic BSM observability at $T = 40$. The four model columns report test macro-F1. The joint loss Δ_{joint} is measured relative to the full deployed 20-feature baseline. The interaction excess is $I_{A,B} = \Delta_{A+B} - (\Delta_A + \Delta_B)$, where the individual losses are taken from Table 19.

Jointly Masked Feature Groups	Masked Features	Dense Macro-F1 (%)	Compressed Macro-F1 (%)	Pruned Macro-F1 (%)	Pruned + Comp. Macro-F1 (%)	Δ_{joint} (pp)	$I_{A,B}$ (pp)
Full 20-feature representation	0	98.85	98.83	98.75	98.67	0.00	—
Pseudonym dynamics + position consistency	3	86.77	86.64	87.00	86.90	11.77	+0.91
Pseudonym dynamics + timing/replay	3	87.06	86.90	87.76	87.54	11.13	+0.90
Zero/constant-position + stop/speed persistence	6	97.60	97.53	97.66	97.65	1.02	+0.25

Using the rounded individual-family losses reported in Table 19, the sums expected under a simple additive interpretation are 10.86, 10.23, and 0.77 points for the three combined scenarios. The measured joint losses exceed these values by approximately 0.91, 0.90, and 0.25 points, respectively. These positive interaction excesses suggest non-additive complementarity between the paired evidence channels. However, because the experiments use a single random seed and the interaction calculation relies on rounded individual losses, the result should be interpreted as evidence of complementarity rather than as a statistically established interaction effect.

4.10. Class-Wise Performance Analysis

Aggregate metrics confirm that the final pruning-plus-compression TensorFlow Lite model preserves strong global performance, but class-wise analysis remains necessary because VeReMi Extension is highly imbalanced and rare attack classes may be hidden by accuracy alone. Therefore, this subsection focuses on the selected deployment-oriented model under $T = 40$ and compares its per-class F1-scores under ideal and realistic BSM observability. The results are summarized in Table 21.

Table 21. Class-wise F1-scores of the final pruning-plus-compression TensorFlow Lite model under ideal and realistic BSM observability for the selected $T = 40$ configuration.

Class	Ideal F1 (%)	Realistic F1 (%)	Gap (pp)
0	99.71	99.56	−0.15
1	99.91	98.99	−0.92
2	89.51	85.53	−3.98
3	100.00	100.00	0.00
4	100.00	100.00	0.00
5	100.00	100.00	0.00
6	99.97	100.00	+0.03
7	100.00	100.00	0.00
8	99.97	100.00	+0.03
9	96.97	94.27	−2.70
10	98.56	98.82	+0.26
11	98.94	99.28	+0.34
12	99.58	99.59	+0.01
13	99.74	99.45	−0.29
14	99.99	99.89	−0.10
15	99.66	99.29	−0.37
16	99.81	99.71	−0.10
17	99.74	99.87	+0.13
18	100.00	99.82	−0.18
19	99.64	99.36	−0.28
Macro average	**99.09**	**98.67**	**−0.42**

Table 21 details the class-wise F1-scores of the retained $T = 40$ compressed model. Under ideal observability, 16 out of 20 classes exceed 99% F1-score. Classes 3, 4, 5, 7, and 18 reach 100%, corresponding respectively to Random Position, Random Position Offset, Constant Speed, Random Speed, and DoS Random Sybil.

The transition to realistic BSM observability reduces the global macro-F1 by only 0.42 percentage points, but the degradation is concentrated in a small number of classes. Class 2 (Constant Position Offset) exhibits the largest decrease, from 89.51% to 85.53%, corresponding to 3.98 percentage points. Class 9 (Eventual Stop) decreases from 96.97% to 94.27%, corresponding to 2.70 percentage points.

For Class 2, this degradation is physically consistent with noisy position observability, which can partially mask fixed spatial offsets. For Class 9, the remaining errors are consistent with overlap between legitimate deceleration or stationary behavior and the transition toward a falsely reported stopped state. Under realistic observability, Class 2 achieves 95.22% precision and 77.63% recall, whereas Class 9 achieves 98.80% precision and 90.14% recall. The degradation of both classes is therefore mainly recall-driven.

At the macro level, the final compressed model decreases from 99.09% macro-F1 under ideal observability to 98.67% under realistic observability, corresponding to a limited gap of 0.42 percentage points. The corresponding realistic macro-precision and macro-recall are 99.45% and 98.00%, respectively. This confirms that the remaining observability-induced degradation is mainly caused by missed detections in a few difficult classes rather than by broad false-positive instability across the taxonomy.

5. Discussion

The results in Section 4 show that the proposed temporal-window-aware physics-informed LSTM achieves strong 20-class misbehavior detection under both ideal and realistic BSM observability. However, classification scores alone do not fully explain the behavior of the system. The main question is how observability realism, temporal context, feature design, compression, and edge deployment interact. This section discusses these issues through five dimensions: BSM observability, the performance–deployment trade-off, the role of physics-informed features, comparison with existing vehicular IDSs, and threats to validity.

5.1. Impact of BSM Observability on IDS Performance

A central objective of this study is to quantify how the controlled BSM observability condition affects multi-class IoV misbehavior detection. Both conditions originate from VeReMi Extension simulation. The ideal regime provides an upper-bound reference based on clean simulator-side kinematics, whereas the simulation-based realistic regime evaluates the same pipeline using noise-inclusive BSM-derived fields reconstructed from the dataset-provided sensor-error components. The latter better imitates receiver-side measurement uncertainty than the ideal condition, but it does not represent real-world V2X validation.

For the selected $T = 40$ pruning-plus-compression TensorFlow Lite model, ideal observability yields 99.60% accuracy and 99.09% macro-F1. Under realistic BSM observability, the same model reaches 99.38% accuracy and 98.67% macro-F1. The accuracy gap is therefore only 0.22 percentage points, and the macro-F1 gap is 0.42 percentage points. This shows that noisy BSM observability degrades performance, but only moderately. The limited degradation suggests that the model does not merely exploit clean numerical precision. Instead, it learns temporal and physical regularities that remain informative when the base kinematic fields are perturbed.

The temporal-window analysis reinforces this conclusion. Under realistic observability, the final model improves from 99.09% accuracy and 97.97% macro-F1 at $T = 10$ to 99.38% accuracy and 98.67% macro-F1 at $T = 40$, indicating that longer sender-local histories help separate persistent or progressive attacks from short-term noise and benign motion variability. Class-wise results further show that most behaviors retain high F1-scores, with residual errors concentrated mainly in ambiguous classes such as 2 and 9 rather than distributed across the full taxonomy.

Overall, the dual-observability protocol quantifies the cost of including simulated measurement uncertainty. The ideal regime should be interpreted as an upper-bound diagnostic reference, whereas the realistic regime is the principal simulation-based noise-robustness reference. Future BSM-based IDS studies should explicitly distinguish clean simulator-side fields, simulated noise-inclusive observables, and measurements collected from physical receivers. Adaptive attackers constitute an additional robustness dimension that is not explicitly modeled in the present evaluation.

5.2. Performance–Deployment Trade-Off

The second major finding is that most of the deployment gain comes from TensorFlow Lite conversion, whereas pruning primarily contributes to the compressed deployment path when combined with this export. As detailed in Section 4.4, moving from the dense Keras baseline to the final Pruning + Compression TensorFlow Lite artifact yields latency speedups above 200× under both observability regimes, while reducing the serialized footprint by more than 86%. This confirms that the main value of the deployment path is not only accuracy retention, but also the conversion of a recurrent IDS into a compact exported artifact with a measurable footprint–latency profile.

This compression behavior is important because recent automotive and time-series compression studies often rely on more elaborate combinations of pruning, quantization, and distillation [17,20]. Such techniques can be highly effective, but they also increase the number of optimization stages and implementation assumptions. In contrast, the present deployment path is intentionally simple: dense training, constant-sparsity pruning-aware fine-tuning, pruning-wrapper stripping, and TensorFlow Lite conversion. The resulting artifact loses only approximately 0.04 accuracy points and 0.11 macro-F1 points relative to the dense ideal baseline, and approximately 0.10 accuracy points and 0.18 macro-F1 points relative to the dense realistic baseline.

The architecture-width ablation further shows that the retained 64/32/32 recurrent stack lies near the knee of the performance–deployment curve. The smaller 32/16/16 configuration reduces footprint and latency but also produces a measurable reduction in validation macro-F1. Conversely, 128/64/64 improves validation macro-F1 by only 0.0019 percentage points while requiring 3.43× more parameters, 2.46× greater TensorFlow Lite footprint, and 2.32× higher controlled-runtime latency than 64/32/32. The intermediate architecture therefore provides the most defensible balance between minority-class performance and edge resource requirements.

These results indicate that the proposed architecture has enough redundancy for lightweight compression without losing its class-balanced diagnostic behavior. The additional C++ single-thread benchmark strengthens this deployment analysis by removing Python batching and interpreter overhead from the timing path. Nevertheless, all latency values should still be interpreted as benchmark-specific measurements of exported artifacts, not as hardware-independent OBU guarantees. The present measurements therefore support the exported artifact's edge-deployment potential, while full hardware-in-the-loop validation remains necessary before production-level OBU or RSU claims can be made.

5.3. Role of Physics-Informed Features

The ablation results in Section 4.9 clarify the role of the physics-informed representation. The compact 20-feature space is not only a numerical enrichment of the raw input. It exposes attack-relevant evidence that is weak or ambiguous in raw kinematics alone.

This point is most visible in the raw-kinematics-only ablation. When the model retains only position, speed, acceleration, and heading, realistic macro-F1 drops from 98.67% to 67.49%. This confirms that instantaneous kinematics are insufficient for robust 20-class diagnosis under realistic BSM observability.

The strongest individual effects come from pseudonym dynamics and position consistency. Pseudonym dynamics capture identity evolution, which is central for Sybil-related and identity-dependent attacks. Position-consistency descriptors capture displacement–speed coherence, which is critical for position-offset, random-position, disruptive, and replay-like behaviors. These two feature families provide evidence that cannot be fully reconstructed from raw kinematic channels alone. This interpretation is consistent with the VeReMi Extension attack taxonomy. Sybil-related classes require evidence about abnormal identity evolution, which explains the strong impact of the pseudonym-dynamics descriptor. In contrast, position-offset, random-position, and disruptive-motion classes depend more directly on displacement–speed coherence, which explains the sensitivity to position-consistency residuals. Replay and DoS-related behaviors also benefit from timing and message-reuse cues, even though their aggregate ablation impact is smaller than that of pseudonym dynamics or position consistency.

This design provides a middle ground between rule-based plausibility checking and purely data-driven deep learning. The descriptors remain interpretable and receiver-side computable, while the recurrent model learns how to combine them across temporal windows. The same observation supports the decision not to use SMOTE or GAN-based balancing: instead of fabricating trajectories that may violate vehicular motion constraints, the framework preserves the original distribution and improves separability through temporal, identity, and consistency cues.

5.4. Comparison with Existing Vehicular IDS Approaches

The proposed IDS differs from existing vehicular intrusion-detection approaches along five dimensions. First, it is evaluated on BSM-centric VeReMi Extension streams rather than on generic network-intrusion or intra-vehicle CAN datasets. Second, it performs unified 20-class diagnosis rather than binary or attack-wise detection. Third, it explicitly separates ideal simulator-side observability from realistic noisy BSM-derived observability. Fourth, it analyzes temporal window length as a deployment parameter. Fifth, it reports a compact TensorFlow Lite artifact with footprint and latency measurements.

These distinctions matter because many existing IDSs optimize only part of the problem. Generic network and CAN datasets are useful for conventional intrusion analysis, but they do not preserve the spatio-temporal semantics of BSM exchange. Binary detectors can identify suspicious behavior, but they do not distinguish attack types that may require different mitigation strategies. Attack-wise evaluations are useful for targeted studies, but they do not show whether a single model can separate the full VeReMi Extension taxonomy. Finally, methods that rely on cloud assistance, federated coordination, or heavy architectures may not directly translate to standalone OBU-oriented inference.

Table 22 focuses on recent IDS frameworks evaluated on the VeReMi Extension dataset or closely related VeReMi Extension settings. Since the dataset dimension is common to the comparison, the table omits a dedicated dataset column and instead emphasizes the factors that make the reported scores heterogeneous: class definitions, preprocessing strategies, observability assumptions, deployment targets, runtime platforms, and evaluation protocols.

The reported values should therefore not be interpreted as a direct ranking, even though all entries are VeReMi-oriented.

Early 20-class and replay-focused VeReMi Extension studies illustrate this heterogeneity. Slama et al. [43] evaluated the same labeled tabular derivative using several conventional machine- and deep-learning classifiers. Their original 20-class Random Forest with recursive feature elimination reported an F1-score of 69.60%; however, its non-temporal tabular protocol and unspecified F1 averaging convention prevent direct comparison with the sender-disjoint sequence-level evaluation adopted here. Slama et al. [18] subsequently addressed the 20-class setting through an ensemble one-versus-all binary-tree decomposition, but this protocol differs from a single unified 20-class softmax classifier and does not report compressed deployment evidence. Moushi et al. [34] focus on data replay and data replay Sybil attacks using reformulated GPS/BSM features, but they do not cover the full 20-class taxonomy or dual BSM observability.

Table 22. Reported quantitative results of recent IDS frameworks evaluated on the VeReMi Extension dataset or closely related VeReMi Extension settings. Values are taken from the respective papers and are not directly comparable because of differences in class definitions, preprocessing strategies, evaluation protocols, observability assumptions, deployment targets, and runtime platforms. (Not reported: NR, Not Tabulated: NT). For Slama et al. [43], the value marked with * is the F1-score reported by the authors; its averaging convention was not explicitly specified and it should therefore not be interpreted as directly equivalent to the macro-F1 reported in the present work.

Reference	Year	Classifier Type	Prep. Augmentation	Classes	Acc (%)	Prec (%)	Rec (%)	Reported F1 Macro-F1 (%)	Model Size	Inference Latency
Slama et al. [43]	2022	Random Forest + RFE	Feature selection (no augmentation)	Multi (20)	NR	71.00	69.10	69.60 *	NR	NR
Slama et al. [18]	2023	Ensemble OVA-BT	Moderate (data split)	20 via OVA	61.00	75.68	78.30	76.80	NR	NR
Alladi et al. [47]	2023	DLCEs with LSTM/CNN–LSTM	Sequence construction from GT files	18	99.63	96.86	96.74	96.75	NR	0.32 ms prediction; 251.56 ms total on Jetson Nano; 337.16 ms total on Raspberry Pi 3B
Moushi et al. [34]	2024	ML ensemble + reformulated features	GPS/BSM feature reformulation	Replay-focused	99.77	NR	NR	99.65	NR	NR
Baharlouei et al. [49]	2024	Federated Learning (ADVENT)	Time slicing	DoS/(D)DoS-oriented	NR	NR	NR	95.28–97.84	NR	NR
Missara et al. [13]	2025	Transformer + FL	Federated learning	Attack-wise	99.37	99.91	98.69	99.30	NR	NR
Kapoor et al. [30]	2025	CNN-LSTM + SMOTE	Heavy (SMOTE)	Binary	98.15	98.50	98.50	98.40	NR	NR
Verma et al. [12]	2025	CatBoost Ensemble	SMOTE + LIME	Multi	92.30	N/A	N/A	89.00	NR	NR
Zeng and Mi [11]	2025	Robust LSTM + FedAware FL	Federated aggregation	Aggregate accuracy only	97.63	NR	NR	NR	NR	NR
Youness et al. [26]	2025	Bi-LSTM + engineered features	Heavy (SMOTE)	Multi (20)	99.59	99.59	NR	99.59	24.8 MB	40.5 ms at 2 M; 41.76 ms at 500 K
Cao et al. [25]	2025	Split learning + Llama3.1-8B	Vehicle–cloud split	Binary anomaly	99.96	100.00	99.92	99.96	568 KB vehicle-side; 14.98 GB cloud LLM	50.22 ms per BSM sequence
Sultana et al. [41]	2025	LA-DETECTS local data-centric checks	Kalman/Hampel + plausibility checks	Attack-wise	NR	NR	NR	NR	NR	NR

Table 22. *Cont.*

Reference	Year	Classifier Type	Prep. Augmentation	Classes	Acc (%)	Prec (%)	Rec (%)	Reported F1 Macro-F1 (%)	Model Size	Inference Latency
Gurjar et al. [15]	2025	RNN-BiLSTM + FL	SMOTE + federated aggregation	Aggregate graphical results	NT	NT	NT	NT	NR	NR
Mahesh et al. [46]	2025	HyperLSTM	Overlapping windows + dropout	Binary	98.00	96.85	98.01	97.41	NR	0.0006 s
Proposed Work	**2026**	**Physics-informed LSTM + TFLite**	**Light (20 features, no SMOTE)**	**Multi (20)**	**99.38**	**99.45**	**98.00**	**98.67**	**88.38 KB**	**3.544 ms (Python CPU) 1.356 ms (C++ single-thread) 0.1283 ms (controlled)**

Several recent works report strong scores while targeting binary or attack-wise tasks. Kapoor et al. [30] collapse VeReMi Extension into a binary normal/attack task before applying SMOTE and CNN–LSTM modeling. Alladi et al. [47] are also directly relevant because they evaluate LSTM and CNN–LSTM Deep Learning Classification Engines on VeReMi Extension and report edge-testbed runtime on Jetson Nano and Raspberry Pi 3B. However, their protocol uses 18 behavior types, overlapping 20-step sequences, and edge-server-oriented inference, rather than the unified 20-class, dual-observability, sender-disjoint, compressed TensorFlow Lite setting considered here. Mahesh et al. [46] also confirm the relevance of recurrent temporal modeling on VeReMi Extension, but their HyperLSTM evaluation remains binary normal/attack detection with overlapping 15-step windows. Sultana et al. [41] propose LA-DETECTS, a local data-centric detector based on Kalman-filter innovation errors, Hampel-filter consistency checks, plausibility rules, and weighted averaging. However, LA-DETECTS reports attack-wise rule-based behavior rather than unified 20-class learned classification.

Federated and distributed approaches should also be interpreted cautiously. Zeng and Mi [11] propose a robust LSTM with FedAware aggregation, but the reported results are mainly aggregate accuracy values without the same macro-F1, class-wise, dual-observability, or deployment-artifact evidence. Gurjar et al. [15] evaluate several recurrent architectures, including RNN, LSTM, BiLSTM, RNN–LSTM, and RNN–BiLSTM, within a Flower-based federated-learning framework. Their study is valuable for analyzing federated recurrent learning under different client counts and aggregation strategies, but it does not provide a standalone TensorFlow Lite artifact, serialized footprint, embedded latency, or explicit ideal-versus-realistic BSM observability comparison. ADVENT should likewise be interpreted as an attack-wise DoS/(D)DoS-oriented evaluation. Baharlouei et al. [49] evaluate attack-onset and malicious-node detection across DoS, DoS Random, DoS Random Sybil, DoS Disruptive, and DoS Disruptive Sybil scenarios. The reported F1 range in Table 22 therefore corresponds to attack-wise ADVENT results rather than to unified 20-class BSM classification.

Among the closest recent baselines, VeMisNet and IoVSL are particularly informative but still differ from the present setting. Youness et al. [26] report strong 20-class VeReMi Extension results using engineered features, SMOTE, and Bi-LSTM modeling, but the model remains in the tens-of-megabytes regime and does not provide a compressed TensorFlow Lite artifact. Cao et al. [25] achieve high binary anomaly-detection performance through split learning, but the inference pipeline depends on a vehicle–cloud architecture and a cloud-side LLM. Verma et al. [12] provide a CatBoost-based multi-class baseline with SMOTE and explainability support, but without temporal sequence modeling, dual observability, or compressed-artifact and runtime evidence. Missara et al. [13] report strong

Transformer-based results in a federated 5G-edge setting, but the evaluation is attack-wise rather than a unified 20-class sequence-level softmax diagnosis.

Taken together, these comparisons show that the proposed IDS should not be read as a score-only competitor to all prior works, because the underlying tasks and deployment assumptions differ substantially. Its main contribution is the joint treatment of fine-grained 20-class diagnosis, dual BSM observability, temporal-window sensitivity, leakage-aware sender-level validation, physics-informed feature-family ablation, and a standalone TensorFlow Lite artifact with measured edge-deployment potential.

5.5. Limitations and Threats to Validity

The results should be interpreted in light of four groups of limitations: internal validity, external validity, deployment validity, and security validity.

5.5.1. Internal Validity

The proposed evaluation uses a fixed random seed, a fixed sender-level split, and a fully specified training and compression pipeline. This makes the experiments reproducible, but it does not provide a full statistical characterization of training stochasticity. Although Section 4.9 strengthens the internal analysis through a training-time feature-family ablation of the retained realistic $T = 40$ configuration, the study does not yet report repeated-seed confidence intervals or sender-level bootstrap intervals. Future work should quantify uncertainty for accuracy, macro-F1, rare-class F1, latency, and ablation effects across multiple random seeds and sender-level resamples.

The selected 30% pruning target should be interpreted as a conservative deployment-oriented configuration rather than as a universally optimal sparsity level. As shown in Table 17, increasing the target to 70% decreases macro-F1 from 99.09% to 98.89% under ideal observability and from 98.67% to 98.56% under realistic observability. Moreover, the additional unstructured sparsity does not reduce the serialized TensorFlow Lite footprint under the current default conversion pipeline. These results support retaining 30% as the better accuracy–sparsity compromise, while structured pruning, layer-wise sparsity allocation, sparsity-aware serialization, and hardware-specific sparse acceleration remain future research directions.

5.5.2. External Validity

Although VeReMi Extension is more suitable for BSM-centric misbehavior detection than generic network-intrusion or CAN datasets, it remains a simulation-based benchmark. The LuST scenario, attack models, mobility traces, and sensor-error assumptions provide a controlled and reproducible evaluation setting, but they cannot fully capture all real V2X deployment conditions. Examples include heterogeneous hardware, changing propagation conditions, localization quality, driver behavior, infrastructure density, and implementation-specific communication stacks.

The sender-level partitioning strategy reduces trajectory leakage by ensuring that no sender appears in more than one split. This is stricter than random message-level splitting. However, it does not guarantee generalization to unseen cities, traffic densities, mobility models, attack implementations, or communication environments. Cross-scenario and cross-dataset validation is therefore required before drawing broader conclusions about external validity.

5.5.3. Deployment Validity

Both the ideal and the simulation-based realistic observability regimes are derived entirely from the VeReMi Extension simulation. The realistic regime improves observability realism by reconstructing each kinematic variable as $q_i^r = q_i + q_i^n$, using the noise-free value

and the associated sensor-error component supplied separately by the dataset. Consequently, it provides a controlled noise-inclusive counterpart to the ideal regime and better imitates receiver-side measurement uncertainty. However, it does not replace validation using BSMs collected from physical vehicles, OBUs, RSUs, or an operational V2X communication stack. Packet loss, communication delay, interference, channel congestion, GNSS outages, synchronization errors, heterogeneous sensor-fusion pipelines, and end-to-end message-processing overhead may affect real-world detection performance.

The TensorFlow Lite artifacts should also not be interpreted as full-integer quantized models. The current conversion uses TensorFlow Lite default optimization and selected TensorFlow operations for recurrent-layer compatibility, but it does not use representative-dataset calibration, integer-only operator constraints, or integer input–output tensors. Future work should evaluate post-training full-integer quantization, quantization-aware training, representative-dataset calibration, operator-level optimization, and hardware-specific acceleration.

The reported latency values are controlled-runtime and CPU-oriented measurements, not production OBU guarantees. The TensorFlow Lite protocol reports mean batch-normalized per-sequence latency under the experimental runtime, and the CPU benchmark is executed on a virtualized Intel Xeon x86-64 platform. These measurements support relative comparison among exported variants and provide an indication of computational cost, but they do not replace hardware-in-the-loop evaluation on ARM-based OBUs, roadside edge devices, or automotive-grade accelerators. Future evaluations should report warm-up policy, median latency, variance, tail latency, memory use, and end-to-end pipeline overhead.

5.5.4. Security Validity

The proposed IDS focuses on closed-set BSM-level classification within the 20-class VeReMi Extension taxonomy. It is therefore a detection component rather than a complete V2X security solution. It does not replace authentication, certificate management, trust management, secure logging, revocation, jamming mitigation, or adversarial robustness mechanisms.

Adaptive attackers are not explicitly modeled in the present evaluation. A strong adversary could craft BSM streams that remain locally consistent in position, speed, acceleration, heading, timing, or pseudonym evolution while still misleading neighboring vehicles. This limitation is important because adversarially optimized or zero-day IoV attacks may exploit learned decision boundaries rather than only violating simple plausibility rules [31]. In white-box conditions, an attacker knowing the feature equations, $T = 40$, the scaling procedure, and the classifier could optimize physically constrained perturbations against the learned boundary. The present work therefore demonstrates strong benchmark-level performance, not certified adversarial robustness.

Finally, the classifier uses a closed-set softmax output. Unknown, zero-day, or adversarially optimized trajectories may therefore be assigned to known classes with high confidence. The current pipeline does not include uncertainty estimation, calibration, Bayesian dropout, ensemble disagreement, or rejection mechanisms. Future work should evaluate physically constrained evasion, adaptive replay and Sybil strategies, out-of-distribution detection, and uncertainty-aware rejection.

6. Conclusions

This paper presented a temporal-window-aware physics-informed edge-oriented IDS for 20-class IoV misbehavior detection under two observability conditions derived entirely from the VeReMi Extension simulation. The ideal regime uses the noise-free kinematic

component q_i, whereas the simulation-based realistic (noise-inclusive) regime reconstructs the observable variable as $q_i^r = q_i + q_i^n$, using the sensor-error component supplied separately by the dataset. The latter provides a controlled approximation of receiver-side measurement uncertainty relative to clean simulator fields; it does not constitute validation using physical vehicles, operational OBUs or RSUs, or real V2X wireless links.

The proposed pipeline combines sender-level leakage-aware splitting, a compact 20-feature physics-informed representation, LSTM-based temporal modeling with residual temporal pooling, and a reproducible pruning-plus-TensorFlow-Lite export path. The temporal sensitivity analysis identifies $T = 40$ as the retained performance–latency compromise in the simulation-based realistic regime while preserving near-peak performance under ideal observability.

Under the retained $T = 40$ configuration, the final pruned-and-compressed TensorFlow Lite artifact achieves 99.60% accuracy and 99.09% macro-F1 under ideal observability, and 99.38% accuracy and 98.67% macro-F1 under simulation-based realistic observability, with an 88.38 KB serialized footprint. Runtime evaluation reports 0.1283 ms controlled-runtime latency, 3.544 ms average latency in the Python-based CPU benchmark, and 1.356 ms average unbatched latency in the single-thread C++ TensorFlow Lite benchmark. These measurements demonstrate compactness and low computational cost under the tested software and CPU conditions and support the artifact's edge-deployment potential; they do not constitute production OBU or RSU performance guarantees.

The architecture-width ablation supports the retained 64/32/32 recurrent configuration. It preserves essentially the same class-balanced validation performance as the substantially larger 128/64/64 model while reducing its TensorFlow Lite footprint from 217.31 to 88.31 KB and its controlled-runtime latency from 0.3347 to 0.1444 ms. The feature-family ablation further confirms that the engineered descriptors are central to the detector: retaining only raw kinematics reduces macro-F1 in the simulation-based realistic regime from 98.67% to 67.49%, while pseudonym dynamics and position-consistency descriptors produce the largest individual degradations.

Future work will extend the evaluation through cross-scenario, cross-density, cross-mobility, and cross-dataset validation. Emerging benchmark extensions such as VeReMi NextGen [59], when available, should be considered to assess robustness under richer traffic scenarios, updated attack models, and more diverse observability conditions. Representative hardware-in-the-loop experiments on ARM-based OBUs, roadside-edge devices, Raspberry Pi, Jetson platforms, or automotive-grade processors are also required to validate end-to-end latency, memory use, energy consumption, and communication-stack overhead. Further work should examine packet loss, channel congestion, GNSS degradation, adaptive adversaries, open-set attacks, hardware-aware quantization, structured pruning, mixed-precision inference, and knowledge distillation.

Author Contributions: Conceptualization, A.B. and A.N.; methodology, A.B.; software, A.B.; validation, A.B.; formal analysis, A.B.; investigation, A.B. and A.N.; resources, A.B.; data curation, A.B.; writing—original draft preparation, A.B., A.N., S.Z., C.T.C. and L.B.; writing—review and editing, A.B., A.N., C.T.C. and S.Z.; visualization, A.B.; supervision, A.B.; project administration, A.B.; funding acquisition, C.T.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by research project MORELLINO (CIPROM/2023/29), funded by "Direcció General de Ciència i Investigació" Generalitat Valenciana-SPAIN, and by Grant PID2025-168451NB-I00 funded by MICIU/AEI/10.13039/501100011033 and FEDER (EU).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The experiments use the labeled tabular derivative of VeReMi Extension distributed through Mendeley Data as `mixalldata_clean.tar.xz`, Version 1, DOI: https://doi.org/10.17632/k62n4z9gdz.1. The downloaded archive has a size of 329,167,264 bytes and SHA-256 checksum `040d88cf6c27fb3b1f92dd784802943e100aab6c729ffdbad8114e0849c3ca03`. It contains the labeled file `mixalldata_clean.csv`. The preprocessing scripts, experimental configurations, run traces, and model-export commands supporting the reported results will be made available in a public repository upon acceptance, or from the corresponding author upon reasonable request.

Acknowledgments: The authors would like to thank the contributors of the VeReMi and VeReMi Extension datasets. The authors acknowledge the use of Google Gemini 3.5 Pro (Google LLC, Mountain View, CA, USA) and ChatGPT 5.6 Plus (OpenAI, San Francisco, CA, USA) solely for assistance in writing and debugging data preprocessing and processing scripts, and for language refinement. All scientific analysis and conclusions remain strictly the authors' independent work.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ARM	Advanced RISC Machine
BSM	Basic Safety Message
C-ITS	Cooperative Intelligent Transport Systems
CAM	Cooperative Awareness Message
CAN	Controller Area Network
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DL	Deep Learning
DoS	Denial of Service
FL	Federated Learning
FN	False Negative
FP	False Positive
GAN	Generative Adversarial Network
GAP	Global Average Pooling
GMM	Gaussian Mixture Model
GNSS	Global Navigation Satellite System
GPU	Graphics Processing Unit
IDS	Intrusion Detection System
IoV	Internet of Vehicles
ITS	Intelligent Transportation Systems
IQR	Interquartile Range
LLM	Large Language Model
LSTM	Long Short-Term Memory
ML	Machine Learning
NNAPI	Android Neural Networks API
OBU	On-Board Unit
OVA	One-Versus-All
RBM	Restricted Boltzmann Machine
RF	Random Forest
RSU	Roadside Unit
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sampling Technique
TFLite	TensorFlow Lite
TFMOT	TensorFlow Model Optimization Toolkit
TP	True Positive

V2X	Vehicle-to-Everything
VAE	Variational Autoencoder
VANET	Vehicular Ad-hoc Network

References

1. Karim, S.M.; Habbal, A.; Chaudhry, S.A.; Irshad, A. Architecture, Protocols, and Security in IoV: Taxonomy, Analysis, Challenges, and Solutions. *Secur. Commun. Netw.* **2022**, *2022*, 1131479. [CrossRef]
2. van der Heijden, R.W.; Lukaseder, T.; Kargl, F. VeReMi: A dataset for comparable evaluation of misbehavior detection in VANETs. In *Proceedings of the Security and Privacy in Communication Networks: 14th International Conference*; SecureComm 2018; Springer: Cham, Switzerland, 2018; pp. 318–337. [CrossRef]
3. Lampe, B.; Meng, W. Intrusion detection in the automotive domain: A comprehensive review. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 2356–2426. [CrossRef]
4. Khezri, E.; Hassanzadeh, H.; Yahya, R.O.; Mir, M. Security Challenges in Internet of Vehicles (IoV) for ITS: A Survey. *Tsinghua Sci. Technol.* **2025**, *30*, 1700–1723. [CrossRef]
5. Haddaji, A.; Ayed, S.; Fourati, L.C. IoV security and privacy survey: Issues, countermeasures, and challenges. *J. Supercomput.* **2024**, *80*, 23018–23082. [CrossRef]
6. Nandy, T.; Md Noor, R.; Kolandaisamy, R.; Idris, M.Y.I.; Bhattacharyya, S. A review of security attacks and intrusion detection in the vehicular networks. *J. King Saud Univ.-Comput. Inf. Sci.* **2024**, *36*, 101945. [CrossRef]
7. Sumner, R.; Eisenhart, B.; Baker, J. *SAE J2735 Standard: Applying the Systems Engineering Process*; Final Report FHWA-JPO-13-046; U.S. Department of Transportation, ITS Joint Program Office: Washington, DC, USA, 2013.
8. DSRC Technical Committee. *On-Board System Requirements for V2V Safety Communications*; Technical Report J29451_202004; SAE International: Warrendale, PA, USA, 2020. [CrossRef]
9. Aljabri, W.; Abdul Hamid, M.; Mosli, R. Enhancing real-time intrusion detection system for in-vehicle networks by employing novel feature engineering techniques and lightweight modeling. *Ad Hoc Netw.* **2025**, *169*, 103737. [CrossRef]
10. Hoang, T.N.; Kim, D. Detecting in-vehicle intrusion via semi-supervised learning-based convolutional adversarial autoencoders. *Veh. Commun.* **2022**, *38*, 100520. [CrossRef]
11. Zeng, B.; Mi, B. Federated Learning-Based Intrusion Detection for Vehicular Networks Using Robust LSTM. In *Proceedings of the 2025 4th International Conference on Artificial Intelligence, Internet of Things and Cloud Computing Technology (AIoTC)*; IEEE: Piscataway, NJ, USA, 2025. [CrossRef]
12. Verma, B.K.; Runthala, A.; Sharma, P.K.; Singh, S.; Garg, D.K.; Kumar, H. Enhancing Cybersecurity in Connected and Autonomous Vehicles: A CatBoost-Based Ensemble Framework for Multi-Class Threat Detection. In *Proceedings of the 2025 IEEE DELCON—International Conference on Recent Smart Technologies in Engineering for Sustainable Development*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. [CrossRef]
13. Missara, R.; Yakan, H.; Kchok, H.; Fajjari, I.; Aitsaadi, N. Misbehavior Detection System in V2X 5G Edge Networks based on Transformer. In *Proceedings of the 2025 Global Information Infrastructure and Networking Symposium (GIIS)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. [CrossRef]
14. Khan, H.; Tejani, G.G.; AlGhamdi, R.; Alasmari, S.; Sharma, N.K.; Sharma, S.K. A secure and efficient deep learning-based intrusion detection framework for the internet of vehicles. *Sci. Rep.* **2025**, *15*, 12236. [CrossRef] [PubMed]
15. Gurjar, D.; Grover, J.; Kheterpal, V.; Vasilakos, A. Federated learning-based misbehavior classification system for VANET intrusion detection. *J. Intell. Inf. Syst.* **2025**, *63*, 807–830. [CrossRef]
16. Qu, Z.; Cai, Z.; Sun, L.; Muhammad, G. PCNA-IDS: An integrated lightweight intrusion detection system in internet of vehicles with federated contrastive learning and differential privacy. *Knowl.-Based Syst.* **2026**, *336*, 115346. [CrossRef]
17. Li, J.; Ma, Y.; Bai, J.; Chen, C.; Xu, T.; Ding, C. A Lightweight Intrusion Detection System with Dynamic Feature Fusion Federated Learning for Vehicular Network Security. *Sensors* **2025**, *25*, 4622. [CrossRef] [PubMed]
18. Slama, O.; Tarhouni, M.; Zidi, S.; Alaya, B. One Versus All Binary Tree Method to Classify Misbehaviors in Imbalanced VeReMi Dataset. *IEEE Access* **2023**, *11*, 135944–135958. [CrossRef]
19. Deng, L.; Li, G.; Han, S.; Shi, L.; Xie, Y. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proc. IEEE* **2020**, *108*, 485–532. [CrossRef]
20. Zhou, X.; Liu, H.; Shi, C.; Liu, J. *Deep Learning on Edge Computing Devices: Design Challenges of Algorithm and Architecture*; Elsevier: Amsterdam, The Netherlands, 2022.
21. Kim, J.; Chang, S.; Kwak, N. PQK: Model Compression via Pruning, Quantization, and Knowledge Distillation. In Proceedings of the Interspeech 2021, Brno, Czechia, 30 August–3 September 2021; pp. 4568–4572. [CrossRef]
22. Kim, J. Quantization Robust Pruning with Knowledge Distillation. *IEEE Access* **2023**, *11*, 28254–28264. [CrossRef]

23. Duan, Y.; Wang, H.; Wang, X.; Yang, Y.; Li, H.; Wang, X.; Chen, S.; Xu, B. Research on Compression Algorithms for Time Series Prediction Models in Source Measure Units Based on Pruning, Quantization, and Knowledge Distillation. In *Proceedings of the 2025 17th International Conference on Communication Software and Networks (ICCSN)*; IEEE: Piscataway, NJ, USA, 2025; pp. 314–321. [CrossRef]
24. Uddin, M.A.; Chu, N.H.; Rafeh, R.; Barika, M. A Scalable Hierarchical Intrusion Detection System for Internet of Vehicles. *IEEE Internet Things J.* **2025**, *12*, 41591–41607. [CrossRef]
25. Cao, J.; Di, X.; Qi, H.; Li, J.; Yang, H.; Hu, P.; Zhao, L. IoVSL: An IoV Anomaly Detection Method Using Split Learning with the Open-Source Large Language Model. In *Proceedings of the IEEE INFOCOM 2025—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. [CrossRef]
26. Youness, N.; Mostafa, A.; Sobh, M.A.; Bahaa, A.M.; Nagaty, K. VeMisNet: Enhanced Feature Engineering for Deep Learning-Based Misbehavior Detection in Vehicular Ad Hoc Networks. *J. Sens. Actuator Netw.* **2025**, *14*, 100. [CrossRef]
27. Jeyaram, G.; Vidhya, V.; Dhanaraj, R.K.; Alabdultif, A. A Lightweight Sequential AI Framework for Real Time Intrusion Detection in Dynamic Vehicular Networks. *Sci. Rep.* **2026**, *16*, 5217. [CrossRef] [PubMed]
28. Kamel, J.; Wolf, M.; van der Heijden, R.W.; Kaiser, A.; Urien, P.; Kargl, F. VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs. In *Proceedings of the ICC 2020—2020 IEEE International Conference on Communications (ICC)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [CrossRef]
29. Haddaji, A.; Ayed, S.; Fourati, L.C.; Boulahia, L.M. Investigation of Security Threat Datasets for Intra-and Inter-Vehicular Environments. *Sensors* **2024**, *24*, 3431. [CrossRef] [PubMed]
30. Kapoor, P.; Prasad, A.; Singh, R.K. Improved Intrusion Detection in VANETs Using SMOTE and Hybrid Deep Learning Models. In *Proceedings of the 2025 International Conference on Intelligent and Secure Engineering Solutions (CISES)*; IEEE: Piscataway, NJ, USA, 2025; pp. 786–790. [CrossRef]
31. Xu, B.; Zhao, J.; Wang, B.; He, G. Detection of zero-day attacks via sample augmentation for the Internet of Vehicles. *Veh. Commun.* **2025**, *52*, 100887. [CrossRef]
32. Choudhary, R.K.; Tembhurney, S.; Barole, K.; Malik, K.; Raut, R.; Joshi, P. Smart Intrusion Detection Systems: A Novel Intrusion Detection Framework by Leveraging SMOTE-ENN with RF and XGBoost. In *Congress on Smart Computing Technologies (CSCT 2024)*; Saraswat, M., Rajan, A., Chakravorty, A., Eds.; Springer: Singapore, 2024; pp. 637–648. [CrossRef]
33. Abdelmaguid, M.A.; Hassanein, H.S.; Zulkernine, M. A VeReMi-based Dataset for Predicting the Effect of Attacks in VANETs. In *Proceedings of the 26th International ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*; MSWiM '23; Association for Computing Machinery: New York, NY, USA, 2023; pp. 175–179. [CrossRef]
34. Moushi, O.M.; Gunawardena, C.; Ye, F.; Hu, R.Q.; Qian, Y. Machine Learning-Based Detection of Data Replay and Data Replay Sybil Attacks for Vehicular Communication Networks. In *Proceedings of the ICC 2024—IEEE International Conference on Communications*; IEEE: Piscataway, NJ, USA, 2024; pp. 5202–5207. [CrossRef]
35. Oucheikh, R.; Fri, M.; Fedouaki, F.; Hain, M. Deep Anomaly Detector Based on Spatio-Temporal Clustering for Connected Autonomous Vehicles. In *Proceedings of the Ad Hoc Networks: Innovations in Smart Cities Applications Edition 4—12th EAI International Conference*; ADHOCNETS 2020; Springer International Publishing: Cham, Switzerland, 2021; pp. 201–212. [CrossRef]
36. Li, X.; Yang, X.; Zhang, Y.; Yang, J.; Chen, Y. An adaptive joint optimization framework for pruning and quantization. *Int. J. Mach. Learn. Cybern.* **2024**, *15*, 5199–5215. [CrossRef]
37. Khan, N.A.; Rafat, A.S. Optimizing Deep Learning Models for Resource-Constrained Environments with Cluster-Quantized Knowledge Distillation. *Eng. Rep.* **2025**, *7*, e70187. [CrossRef]
38. Yang, L.; Moubayed, A.; Shami, A. MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles. *IEEE Internet Things J.* **2022**, *9*, 616–632. [CrossRef]
39. Korium, M.S.; Saber, M.; Beattie, A.; Narayanan, A.; Sahoo, S.; Nardelli, P.H. Intrusion detection system for cyberattacks in the Internet of Vehicles environment. *Ad Hoc Netw.* **2024**, *153*, 103330. [CrossRef]
40. Chouchene, I.; Eltaief, H.; Youssef, H. Advances in machine and deep learning for intrusion detection in VANETs: A comprehensive survey. *Clust. Comput.* **2026**, *29*, 143. [CrossRef]
41. Sultana, R.; Grover, J.; Tripathi, M.; Sharma, P. LA-DETECTS: Local and Adaptive Data-Centric Misbehavior Detection Framework for Vehicular Technology Security. *IEEE Open J. Veh. Technol.* **2025**, *6*, 145–169. [CrossRef]
42. Slama, O.; Alaya, B.; Zidi, S.; Tarhouni, M. Comparative Study of Misbehavior Detection System for Classifying Misbehaviors on VANET. In *Proceedings of the 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT)*; IEEE: Piscataway, NJ, USA, 2022; pp. 243–248. [CrossRef]
43. Slama, O.; Alaya, B.; Zidi, S. Towards Misbehavior Intelligent Detection Using Guided Machine Learning in Vehicular Ad-hoc Networks (VANET). *Intel. Artif.* **2022**, *25*, 138–154. [CrossRef]
44. Christy, C.; Nirmala, A.; Teena, A.M.O.; Amali, A.I. Machine learning based multi-stage intrusion detection system and feature selection ensemble security in cloud assisted vehicular ad hoc networks. *Sci. Rep.* **2025**, *15*, 27058. [CrossRef] [PubMed]

45. Ahsan, S.I.; Legg, P.; Alam, S.I. An explainable ensemble-based intrusion detection system for software-defined vehicle ad-hoc networks. *Cyber Secur. Appl.* **2025**, *3*, 100090. [CrossRef]
46. Mahesh, T.R.; Gangadevi, G.; Mishra, K.K.; Algarni, A.; Saidani, O.; Kumar, A. HyperLSTM for Anomaly Detection Model for VANET Security Using the VeReMi Extension Dataset. *Int. J. Distrib. Sens. Netw.* **2025**, *2025*, 6789771. [CrossRef]
47. Alladi, T.; Kohli, V.; Chamola, V.; Yu, F.R. A deep learning based misbehavior classification scheme for intrusion detection in cooperative intelligent transportation systems. *Digit. Commun. Netw.* **2023**, *9*, 1113–1122. [CrossRef]
48. Meegammana, N.W.; Fernando, H. Robust DoS Attack Detection for V2X Networks Using RNN Variants with Attention. In *Proceedings of the 2025 7th International Conference on Advancements in Computing (ICAC)*; IEEE: Piscataway, NJ, USA, 2025; pp. 1–6. [CrossRef]
49. Baharlouei, H.; Makanju, A.; Zincir-Heywood, N. Evaluating the Robustness of ADVENT on the VeReMi-Extension Dataset. In *Proceedings of the 2024 20th International Conference on Network and Service Management (CNSM)*; IEEE: Piscataway, NJ, USA, 2024; pp. 1–7. [CrossRef]
50. Nguyen, T.P.; Cho, J.; Kim, D. Semi-supervised intrusion detection system for in-vehicle networks based on variational autoencoder and adversarial reinforcement learning. *Knowl.-Based Syst.* **2024**, *304*, 112563. [CrossRef]
51. Abidi, R.; Ben Azzouna, N.; Trojet, W.; Hoblos, G.; Sahli, N. A study of mechanisms and approaches for IoV trust models requirements achievement. *J. Supercomput.* **2024**, *80*, 4157–4201. [CrossRef]
52. Huang, K.; Xian, R.; Xian, M.; Wang, H.; Ni, L. A comprehensive intrusion detection method for the internet of vehicles based on federated learning architecture. *Comput. Secur.* **2024**, *147*, 104067. [CrossRef]
53. Almansour, S.; Yadav, K.; Alkwai, L.M.; Alghamdi, N.S.; Viriyasitavat, W.; Dhiman, G. Adaptive personalized federated learning with lightweight depthwise convolutional bottleneck network for novel intrusion detection system in internet of vehicles. *Sci. Rep.* **2025**, *15*, 35604. [CrossRef] [PubMed]
54. Campos, E.M.; Gonzalez-Vidal, A.; Hernández-Ramos, J.L.; Skarmeta, A. Federated learning for misbehaviour detection with variational autoencoders and Gaussian mixture models. *Int. J. Inf. Secur.* **2025**, *24*, 95. [CrossRef]
55. Xie, N.; Zhang, C.; Yuan, Q.; Kong, J.; Di, X. IoV-BCFL: An intrusion detection method for IoV based on blockchain and federated learning. *Ad Hoc Netw.* **2024**, *163*, 103590. [CrossRef]
56. Luo, F.; Wang, J.; Li, Z.; Luo, C. An anomaly detection model for in-vehicle networks based on lightweight convolution with spectral residuals. *Comput. Secur.* **2025**, *151*, 104304. [CrossRef]
57. Slama, O.; Tarhouni, M.; Zidi, S.; Alaya, B. VeReMi_Extension: Dataset for Misbehaviors in VANETs. *Mendeley Data* **2023**. [CrossRef]
58. Zhong, H.; Yang, F.; Wei, L.; Zhang, J.; Gu, C.; Cui, J. Dataset for Evaluation of DDoS Attacks Detection in Vehicular Ad-Hoc Networks. In *Proceedings of the Wireless Algorithms, Systems, and Applications*; Wang, L., Segal, M., Chen, J., Qiu, T., Eds.; Springer Nature: Cham, Switzerland, 2022; pp. 249–260. [CrossRef]
59. Hermann, A.; Remmers, J.N.; Eisermann, D.; Erb, B.; Kargl, F. VeReMi NextGen: A Dataset for Evaluating Misbehavior Detection Systems in VANETs. In *Proceedings of the 2026 IEEE Vehicular Networking Conference (VNC), Montreal, QC, Canada, 8–10 June 2026*; IEEE: Piscataway, NJ, USA, 2026.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.