LITE-KAN: Efficient Bit-level Feature Extraction via Linear-equivalent Inference-Time Encoding for In-vehicle Networks

Weipeng Zeng
Computing and Data Science
Xiamen University Malaysia
Sepang, Malaysia
clauvio.weipengzeng@outlook.com

Jing Li*
Computing and Data Science
Xiamen University Malaysia
Sepang, Malaysia
jing.li@xmu.edu.my

Congpu Wang
Computing and Data Science
Xiamen University Malaysia
Sepang, Malaysia
hugh.wangcongpu@outlook.com

***Abstract*—As the Controller Area Network (CAN) used in modern vehicles is subject to increasing attacks, and the end-user resources are too limited to deploy heavyweight intrusion detection systems (IDS), this paper proposes LITE-KAN, a novel lightweight bit-level CAN IDS framework. LITE-KAN trains a constrained Kolmogorov-Arnold Networks (KAN)-style additive encoder and deploys it in an exactly equivalent affine form of a binary input feature extractor, after which it performs classification using a Random Forest. Experiments on the Car-Hacking and CICIoV2024 datasets show that the deployed encoding is highly similar to the original encoder to numerical accuracy, saves up to 6.4 to 9.8 times the storage space of the feature extractor, and speeds up CPU feature extraction by 12 to 43 times. In addition, under 2-dimensional compression, LITE-KAN achieves accuracy rates of 99.09% and 99.85% on the Car-Hacking and CICIoV2024 datasets, respectively, using an RF classifier.**

Keywords: Controller Area Network; Feature Extraction; Internet of Vehicles; Machine Learning.

I. Introduction

To support sensing, control, diagnostics, and communications in modern vehicles, in-vehicle networks are now indispensable, and CAN remains the dominant medium because of its practical deployment advantages [1]. However, CAN was not designed with authentication in mind, which leaves CAN traffic vulnerable to attack and makes intrusion detection essential in practice [2–5]. A major difficulty is that IDS deployment is often resource-constrained: the detector may need to run on ECUs with tight limits on latency, memory footprint, and implementation complexity [6, 7]. Under such conditions, feature extraction becomes a bottleneck when online inference repeatedly evaluates nonlinear basis functions.

This work was supported by the Xiamen University Malaysia Research Fund (XMUMRF), Cycle 17/2026, under Grant XMUMRF/2026-C17/IECE/0071.

The central issue is therefore feature representation. Handcrafted CAN features are easy to use, but they depend on heuristic knowledge of specific messages and may overlook useful information [5]. Learned features from MLPs, CNNs, RNNs, and related architectures can be more discriminative [5, 8], yet their benefits usually come with higher computational cost from nonlinear operations. The objective here is not merely to reduce parameter count, but to retain the benefits of learned representations while simplifying deployment-time computation.

This possibility arises from the binary nature of CAN frames. After converting identifiers and payloads into bit-level form, every input dimension lies in {0,1}. If the retained feature extractor is further constrained to be additive, then each univariate branch is fully determined on this binary domain and can be written explicitly as a linear term in the input. Under these restricted conditions, the retained feature extractor is therefore an affine operator. The point is not to claim a stronger deployment-time function class, but to identify an exactly equivalent and simpler deployment form under the restricted domain considered.

A reasonable question arises: why use a nonlinear architecture if the retained function is affine anyway? The answer is that nonlinear structure can still help during training, even when the retained mapping is affine on a restricted binary domain. In practice, a nonlinear additive architecture can offer a more flexible optimization path for learning compact supervised representations. However, we do not claim that nonlinear training always dominates affine training; that would require direct empirical validation.

Based on this insight, we propose Linear-equivalent Inference-Time Extractor KAN (LITE-KAN), a lightweight CAN IDS framework built on KAN-style additive representations learned from bit-level CAN frames. LITE-KAN uses nonlinear training

with a constrained KAN-style additive encoder, then converts the retained feature extractor into an exact affine deployment form, whose coefficients can be obtained directly or through probing-based recovery. Online inference then reduces to matrix multiplication followed by a Random Forest (RF) classifier. This result is specific to a restricted setting: it is not a general simplification of unconstrained KANs, and this work is not intended to replace CNNs, RNNs, or MLPs. It applies only when the retained mapping is additive; exact conversion does not hold once cross-bit interactions, retained normalization layers, or post-layer activations are included.

II. Related Works

Literature related to this work spans CAN IDS design, lightweight IDS design under ECU constraints, and Kolmogorov-Arnold Networks (KAN) or KAN-inspired architectures. CAN intrusion detection includes traditional machine learning and deep learning approaches [5, 8]; although CNNs, RNNs, MLPs, and related models can learn representations, ECU deployment limits remain challenging [6, 7]. Manually defined features based on identifiers, payload statistics, and timing characteristics therefore remain common [5, 8]. This work instead exploits bit-level binary inputs and a strictly additive encoder to learn low-dimensional representations that can be rewritten exactly as an affine operator for deployment, provided there are no cross-bit interactions, retained normalization layers, or post-layer activations. Lightweight IDS studies use compression, shallow models, offloading, and efficient detectors [6, 7]. KANs use additive learnable univariate functions [9], and FastKAN reduces computational burden [10]; however, standard KAN/FastKAN inference still requires nonlinear basis-function evaluation, so this solution is applicable only to a constrained setting. The idea relates to structural re-parameterization [11], but differs because the exact affine conversion follows from CAN bit-level representation and the strictly additive structure. Thus, unlike prior work, this paper simplifies the feature extractor itself at deployment time.

III. Methodology

A. Overall Framework

LITE-KAN is designed to combine nonlinear supervised representation learning during training with lightweight non-neural inference during deployment. Let a CAN frame be represented as a binary vector $\mathbf{x} \in \{0,1\}^d$, where d denotes the encoded bit length. During training, a structurally constrained FastKAN encoder maps $\mathbf{x}$ to a latent representation $\mathbf{h}(\mathbf{x}) \in \mathbb{R}^m$, where m is the latent feature dimension, and is optimized jointly with a temporary softmax classifier. After training, the softmax head is discarded, and only the retained encoder is kept. Rather than deploying the nonlinear encoder directly, LITE-KAN exploits the binary nature of the input space and rewrites the retained mapping exactly as an affine transformation on $\{0,1\}^d$, as detailed in Section III-C. The overall training-to-deployment pipeline of LITE-KAN is illustrated in Fig. 1.

During deployment, each CAN frame is first mapped into the latent space by this affine operator, and the resulting feature vector is then classified by a lightweight RF. In this way, nonlinear function learning is confined to the training stage, while deployment reduces to a purely non-neural pipeline.

B. Additive Bit-Level Feature Transformation

Given a bit-level input $\mathbf{x} = (x_1, \ldots, x_d)$, the retained encoder is defined additively as

$$h_j(\mathbf{x}) = \sum_{i=1}^{d} g_{j,i}(x_i) + b_j, \quad j = 1, \ldots, m, \tag{1}$$

where $g_{j,i}$ denotes the learned univariate response associated with input bit i and latent dimension j, and b_j is a bias term.

In the present study, the transformation is implemented using a structurally constrained FastKAN layer [10]. Exact affine rewrite on the binary domain requires that the retained feature-extraction layer remain bit-wise additive, without pre-branch cross-dimensional mixing, retained normalization, or post-layer activation. Under these constraints, the retained mapping admits an exact affine rewrite after training. The nonlinear branch parameterization is used only during optimization and should not be interpreted as implying that the deployed function class is strictly richer in the restricted setting considered here.

Training is performed jointly with a temporary softmax classifier. Given labeled samples $\{(\mathbf{x}^{(n)}, y^{(n)})\}_{n=1}^{N}$, the model is optimized using weighted cross-entropy:

$$\mathcal{L} = -\sum_{n=1}^{N}\sum_{c=1}^{C} w_c \, \mathbf{1}[y^{(n)} = c] \log p_c^{(n)}, \tag{2}$$

where $p_c^{(n)}$ is the predicted probability for class c, and w_c is the class weight used to mitigate class imbalance.

C. Exact Affine Rewrite on the Binary Domain

LITE-KAN exploits bit-level CAN inputs, where each coordinate satisfies $x_i \in \{0,1\}$. Under the retained strictly additive encoder, each latent feature sums univariate responses over individual bits. Since

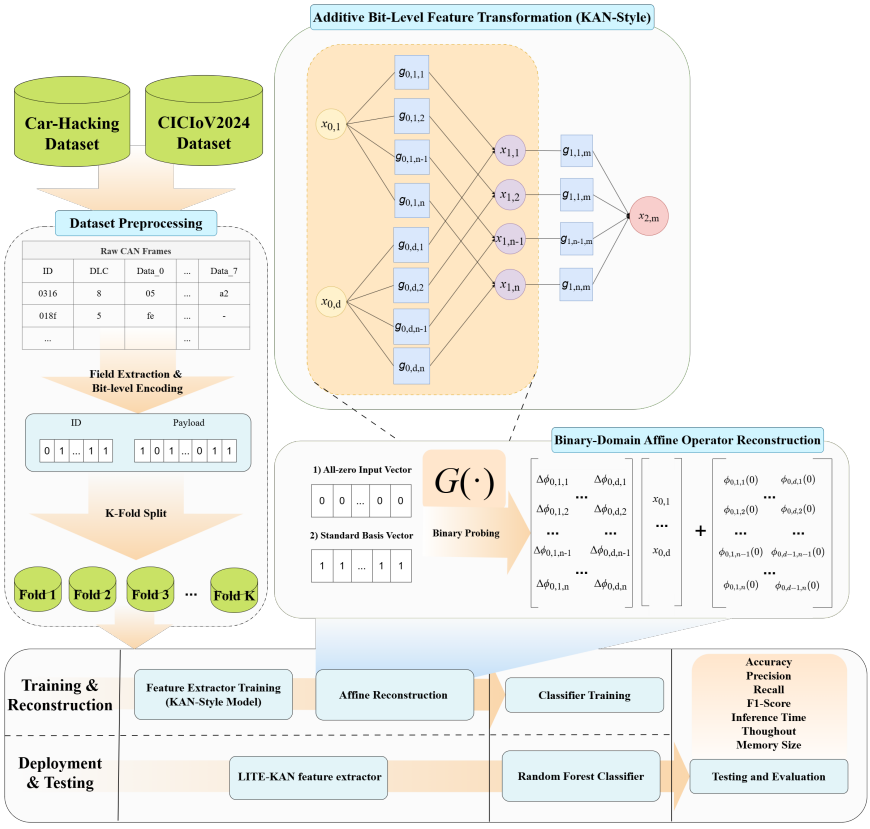


Fig. 1: Overview of LITE-KAN. Training uses a constrained FastKAN encoder, while deployment uses its equivalent affine form with a Random Forest classifier.

a binary univariate response is fully determined by its values at 0 and 1, for any $x_i \in \{0, 1\}$,

$$g_{j,i}(x_i) = (g_{j,i}(1) - g_{j,i}(0))\, x_i + g_{j,i}(0) \tag{3}$$

Substituting this identity into the additive encoder gives the deployment parameters

$$W_{j,i} = g_{j,i}(1) - g_{j,i}(0) \tag{4}$$

$$b'_j = \sum_i g_{j,i}(0) + b_j \tag{5}$$

and therefore

$$h(x) = Wx + b' \tag{6}$$

This rewrite is exact rather than approximate, provided that the retained feature extractor remains strictly additive across input bits, with no normalization / post-layer activation / cross-bit interactions. This result is specific to the restricted setting considered here and should not be interpreted as a general simplification claim for unconstrained KANs.

D. Probing-Based Recovery of W and b'

The affine form can be recovered by probing the trained retained encoder, without inspecting internal branch parameters. Let 0 denote the all-zero input and e_i the i-th standard basis vector. The bias and each column of W are obtained as

$$b' = h(0) \tag{7}$$

$$W_{(:,i)} = h(e_i) - h(0) \tag{8}$$

This probing-based recovery requires $d + 1$ forward evaluations, is performed once after training, and introduces no additional online inference cost.

E. Deployment and Complexity

After the rewrite, feature extraction reduces to

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b}', \tag{9}$$

followed by a compact RF classifier.

Let B denote the number of basis functions in each training-time univariate branch. The original additive KAN-style encoder requires $\mathcal{O}(dmB)$ computation for feature extraction, whereas the rewritten deployment form requires only $\mathcal{O}(dm)$. The deployed extractor stores only $\mathbf{W} \in \mathbb{R}^{m \times d}$ and $\mathbf{b}' \in \mathbb{R}^m$, leading to an $\mathcal{O}(dm)$ storage cost.

IV. Experiments

A. Datasets and Preprocessing

We evaluate on two public CAN intrusion detection datasets: Car-Hacking and CICIoV2024 [12, 13]. Car-Hacking is treated as a 5-class frame-level task, while CICIoV2024 is treated as a 6-class frame-level task using the `specific_class` field. The class distribution is moderately imbalanced in Car-Hacking, where Normal accounts for 29.78%, RPM for 19.72%, gear for 17.99%, DoS for 17.69%, and Fuzzy for 14.81%. In contrast, CICIoV2024 is highly imbalanced, with BENIGN comprising 86.90% of samples, followed by DoS at 5.30%, RPM at 3.90%, SPEED at 1.77%, STEERING_WHEEL at 1.42%, and GAS at 0.71%. Each frame is encoded as a 75-bit vector formed by concatenating the 11-bit identifier and 64 payload bits. Zero-variance dimensions are removed where applicable. We also exclude conflicting-label frame patterns, because the frame-level formulation assumes a deterministic mapping from static frame content to labels.

B. Experimental Settings

All experiments use stratified 10-fold cross-validation in a CPU-only environment. In each fold, the feature extractor is trained on the training split only and, when applicable, converted to its deployment form before generating latent features for both training and test sets. The downstream RF is then trained on the training-fold features and evaluated on the corresponding test fold, and reported results are averaged across folds. To compare feature extraction under a common classification backend, we evaluate raw 75-bit RF, PCA, PLS-DA, a linear bottleneck (LB), MLP, FastKAN, and LITE-KAN. Unless otherwise noted, the latent dimension m varies from 2 to 16. We report accuracy, macro-precision, macro-recall, and macro-F1, and for rewrite fidelity we also measure MAE, maximum absolute error, and MSE. Deployment efficiency is evaluated by storage footprint and CPU-based feature extraction latency.

C. Evaluation Protocol

The experimental analysis was designed to assess the proposed framework from several complementary perspectives, each tied to a distinct practical question. One concern was whether the affine deployment form could reproduce the retained FastKAN encoder with negligible numerical discrepancy, thereby establishing that the post-training conversion preserves the learned transformation rather than merely approximating it. Another was whether this rewrite leads to tangible deployment benefits, particularly in terms of storage reduction and lower CPU inference cost relative to the original FastKAN extractor.

Performance under aggressive dimensionality reduction was also of central interest, since the intended application of LITE-KAN is not only lightweight deployment but also compact supervised representation learning. In that setting, comparison with conventional linear methods and neural baselines helps reveal whether the rewritten features preserve discriminative information when the latent space is severely constrained. Particular emphasis was placed on the 2-dimensional case, where compression is strongest and differences in representation quality become especially visible. The comparison with the raw 75-bit RF baseline further provides an end-to-end view of whether a highly compressed learned representation can remain competitive with classification performed directly on the original input space.

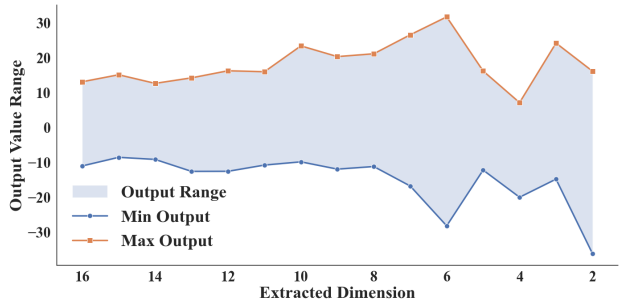
V. Results

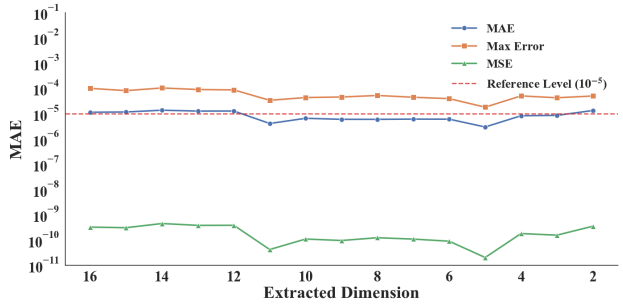
A. Rewrite Fidelity and Deployment Efficiency

Across latent dimensions from 2 to 16, the affine deployment form reproduces the encoder up to floating-point precision. The MAE remains on the order of 10^{-6}–10^{-5}, the maximum absolute error on the order of 10^{-5}–10^{-4}, and the MSE typically on the order of 10^{-11}–10^{-10}. Over the observed encoder output range of approximately $[-44, 35]$, the maximum reconstruction error stays below 1.9×10^{-4}, as shown in Fig. 2. Deployment efficiency also improves substantially: the affine representation requires only 0.59–4.75 KB, compared with 5.8–30.5 KB for the original FastKAN extractor, giving a 6.4–9.8-fold reduction. Feature extraction latency decreases from 360–550 ms to 10–36 ms, corresponding to a 12–43-fold speedup.

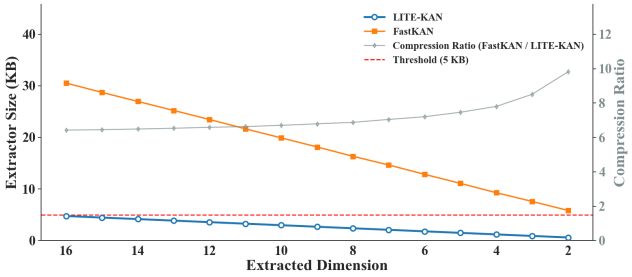
B. Detection Performance Under Low-Dimensional Compression

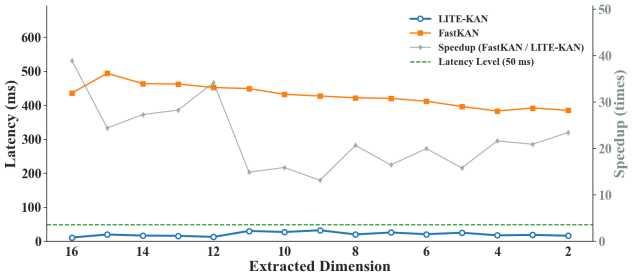
Tables I and II summarize performance at 2 dimensions and the best results across the evaluated latent dimensions, while Fig. 3 shows the trends. Under this aggressive compression, LITE-KAN remains close to the original FastKAN, reaching 99.09% accuracy on Car-Hacking and 99.85% on CICIoV2024 in the 2-dimensional setting. It also outperforms PCA and PLS-DA and remains competitive with other baselines, indicating that the learned compact representation is more informative than conventional linear projections. Compared with the tested MLP baseline, LITE-KAN shows more stable behavior, especially on CICIoV2024, where the MLP exhibits substantial degradation. These results support the intended interpretation: the value of LITE-KAN mainly comes from nonlinear structured parameterization

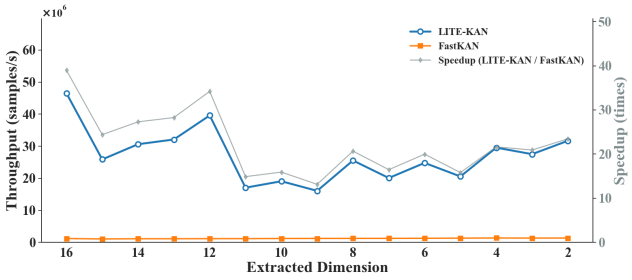




(a) Encoder output range and reconstruction error analysis.



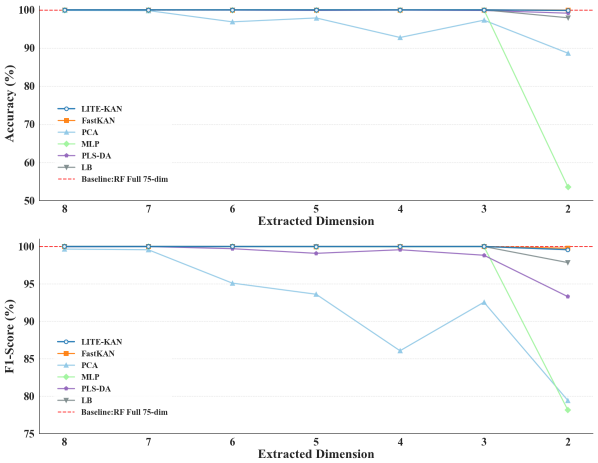




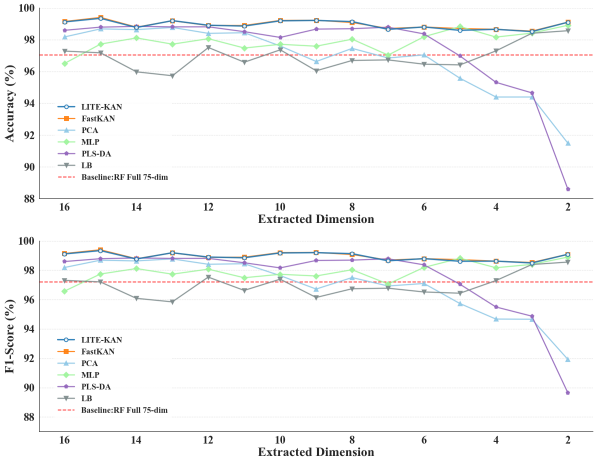
(b) Deployment efficiency in storage, latency, and throughput.

Fig. 2: Rewrite fidelity and deployment efficiency of the affine deployment form.

during training, not from a stronger deployment-time function class, which is affine.



(a) CICIoV2024.



(b) Car-Hacking.

Fig. 3: Detection performance across latent dimensions.

C. End-to-End Comparison with the Raw-Input Baseline

On Car-Hacking, we further compare the same RF classifier trained on raw 75-bit inputs and on 2-dimensional LITE-KAN features. As shown in Fig. 4, the RF using LITE-KAN features maintains competitive detection performance despite operating on a much lower-dimensional affine-rewritten representation.

VI. Discussion and Limitations

Although the deployed feature extractor is affine on the binary input domain, retaining a nonlinear additive parameterization during training remains

TABLE I: Performance on Car-Hacking at 2 dimensions and the best result over multiple runs at 2 dimensions is reported.

Method	Acc. (%)	Macro-F1 (%)	Best Acc. (%)	Best Macro-F1 (%)
PCA	91.49 ± 0.06	91.93 ± 0.05	91.62	92.04
PLS-DA	88.60 ± 0.08	89.65 ± 0.07	88.71	89.74
LB	98.56 ± 0.13	98.55 ± 0.13	98.70	98.69
MLP	98.90 ± 0.02	98.89 ± 0.02	98.93	98.92
FastKAN	99.10 ± 0.14	99.09 ± 0.13	99.27	99.26
LITE-KAN	99.09 ± 0.10	99.08 ± 0.10	99.22	99.21

TABLE II: Performance on CICIoV2024 at 2 dimensions and the best result over multiple runs at 2 dimensions is reported.

Method	Acc. (%)	Macro-F1 (%)	Best Acc. (%)	Best Macro-F1 (%)
PCA	88.66 ± 0.94	79.44 ± 0.76	89.39	80.57
PLS-DA	99.14 ± 0.17	93.30 ± 1.11	99.40	95.01
LB	97.93 ± 0.13	97.85 ± 0.12	97.99	97.97
MLP	53.65 ± 2.11	78.16 ± 0.42	57.80	78.97
FastKAN	99.94 ± 0.09	99.77 ± 0.30	100.00	100.00
LITE-KAN	99.85 ± 0.23	99.57 ± 0.76	100.00	100.00

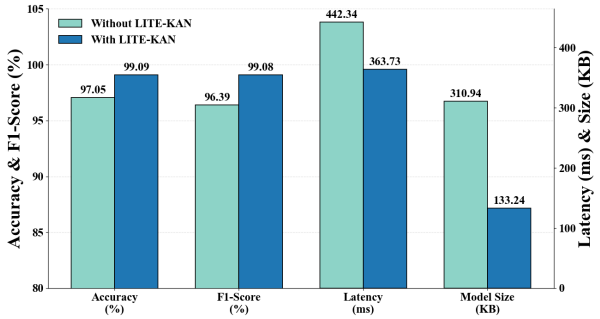


Fig. 4: RF on raw 75-bit inputs versus RF on 2-dimensional LITE-KAN features.

useful because it provides a more flexible optimization path for learning compact supervised representations from bit-level CAN inputs. This matters especially in the low-dimensional regime, where directly optimizing a small linear projection can be too restrictive to preserve task-relevant discriminative structure. The results support this interpretation: after the exact rewrite, LITE-KAN remains close to the original FastKAN representation while staying competitive with or better than conventional linear compression baselines under strong dimensionality reduction.

From a deployment perspective, the main benefit of LITE-KAN is that it simplifies runtime feature extraction itself. After conversion, inference reduces to matrix-vector multiplication and bias addition, followed by a compact RF classifier. This removes repeated nonlinear basis-function evaluation at deployment time and yields a simpler execution path that is better aligned with resource-constrained environments such as automotive ECUs.

The limitations are equally important. First, the exact affine rewrite holds only under a restricted setting: bit-level binary inputs and a retained encoder that is strictly additive across input bits, without cross-bit interactions, retained normalization, or post-layer activation. Second, the current formulation is frame-level and does not model temporal dependencies, message ordering, or timing behavior. Third, deployment efficiency has been validated only in a CPU-only environment, not on production-grade automotive ECUs, and evaluation is limited to two public datasets.

VII. Conclusion

LITE-KAN provides a deployment-oriented feature-extraction framework for frame-level CAN intrusion detection in which nonlinear training and affine deployment are cleanly separated. Under a bit-level binary input representation and an additive retained encoder structure, the learned transformation can be rewritten exactly on $\{0,1\}^d$ as an affine map, allowing a nonlinear additive parameterization during training and an equivalent affine transformation at deployment, followed by a lightweight RF classifier.

The main contribution is not a stronger deployment-time function class, but a useful decoupling between training-time parameterization and deployment-time computation made possible by explicit structural constraints. Experiments on Car-Hacking and CICIoV2024 show that this exact affine deployment form matches the retained encoder to numerical precision

while reducing deployment cost and maintaining competitive performance under low-dimensional compression. The method remains restricted to binary-domain additive feature extractors and does not address temporal CAN attack modeling or retained layers with cross-bit interactions.

References

[1] Kun Huang, Rundong Xian, Ming Xian, Huimei Wang, and Lin Ni. A comprehensive intrusion detection method for the internet of vehicles based on federated learning architecture. *Computers & Security*, 147:104067, 2024. doi: 10.1016/j.cose.2024.104067.

[2] Emad Aliwa, Omer Rana, Charith Perera, and Peter Burnap. Cyberattacks and countermeasures for in-vehicle networks. *ACM Computing Surveys*, 54(1):1–37, 2022. doi: 10.1145/3431233.

[3] Mahdi Dibaei, Xi Zheng, Kun Jiang, Robert Abbas, Shigang Liu, Yuexin Zhang, Yang Xiang, and Shui Yu. Attacks and defences on intelligent connected vehicles: a survey. *Digital Communications and Networks*, 6(4):399–421, 2020. doi: 10.1016/j.dcan.2020.04.007.

[4] Wufei Wu, Renfa Li, Guoqi Xie, Jiyao An, Yang Bai, Jia Zhou, and Keqin Li. A survey of intrusion detection for in-vehicle networks. *IEEE Transactions on Intelligent Transportation Systems*, 21(3):919–933, 2020. doi: 10.1109/TITS.2019.2908074.

[5] Sampath Rajapaksha, Harsha Kalutarage, M. Omar Al-Kadri, Andrei Petrovski, Garikayi Madzudzo, and Madeline Cheah. Ai-based intrusion detection systems for in-vehicle networks: a survey. *ACM Computing Surveys*, 55 (11):237:1–237:40, 2023. doi: 10.1145/3570954.

[6] Ruth Cordova-Cardenas, Daniel Amor, and Álvaro Gutiérrez. Edge ai in practice: A survey and deployment framework for neural networks on embedded systems. *Electronics*, 14(24):4877, 2025. doi: 10.3390/electronics14244877.

[7] Zhaoyun Zhang and Jingpeng Li. A review of artificial intelligence in embedded systems. *Micromachines*, 14(5):897, 2023. doi: 10.3390/mi14050897.

[8] A hybrid deep learning based intrusion detection system using spatial-temporal representation of in-vehicle network traffic. *Vehicular Communications*, 35:100471, 2022. doi: 10.1016/j.vehcom.2022.100471.

[9] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark. Kan: Kolmogorov-arnold networks. http://arxiv.org/abs/2404.19756, 2025.

[10] Ziyao Li. Kolmogorov-arnold networks are radial basis function networks. http://arxiv.org/abs/2405.06721, 2024.

[11] Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13728–13737, 2021. doi: 10.1109/CVPR46437.2021.01352.

[12] Eunbi Seo, Hyun Min Song, and Huy Kang Kim. Gids: Gan based intrusion detection system for in-vehicle network. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pages 1–6, 2018. doi: 10.1109/PST.2018.8514157.

[13] Euclides Carlos Pinto Neto, Hamideh Taslimasa, Sajjad Dadkhah, Shahrear Iqbal, Pulei Xiong, Taufiq Rahman, and Ali A. Ghorbani. Ciciov2024: advancing realistic ids approaches against dos and spoofing attack in iov can bus. *Internet of Things*, 26:101209, 2024. doi: 10.1016/j.iot.2024.101209.

Appendix

A. Software and Hardware

All experiments are conducted in a CPU-only environment with the setup show in Table III:

TABLE III: Software and hardware configuration.

Item	Specification
Operating system	Windows 10
CPU	AMD Ryzen 7 5800H
RAM	16 GB
Python	3.12.7
PyTorch	2.10.0
scikit-learn	1.8.0

All experiments use stratified 10-fold cross-validation. In each fold, the additive encoder is first trained, then rewritten into its affine form, and finally paired with a Random Forest classifier trained on the resulting latent features.

B. Training Configuration

The retained additive encoder is implemented using a structurally constrained FastKAN layer and trained with:

TABLE IV: Training configuration for the retained additive encoder.

Hyperparameter	Value
Optimizer	Adam
Learning rate	0.001
Batch size	4096
Max epochs	10
Loss function	Weighted cross-entropy

The downstream classifier is a compact Random Forest (RF) with 50 trees, a maximum depth of 5, the Gini impurity splitting criterion, and balanced class weights.

All experiments use stratified 10-fold cross-validation. In each fold, the additive encoder is first trained, then rewritten into its affine form, and finally paired with the RF classifier trained on the resulting latent features.

C. Recovery Procedure

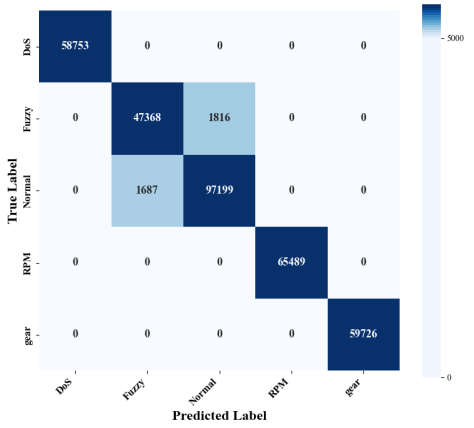
After training, the temporary softmax head is discarded. The deployment-time affine parameters W and b' are recovered from the retained encoder using the probing-based recovery procedure in Section III-D. This procedure requires $d+1$ forward evaluations and is performed once after training.

Figure 3 presents the accuracy and macro-F1 trends as the latent dimension is reduced. On Car-Hacking, LITE-KAN remains close to FastKAN across the evaluated dimensions, with only limited degradation under stronger compression. A similar tendency is observed on CICIoV2024. These results further support the claim that the affine deployment rewrite preserves most of the discriminative structure learned by the nonlinear encoder.

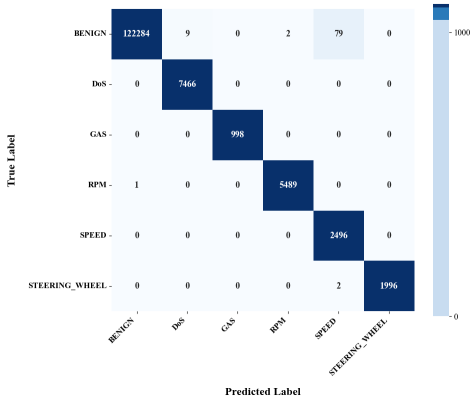
In addition, Fig. 5 provides a 2-dimensional visualization of LITE-KAN results for both datasets.

To support reproducibility, part of the source code for the proposed LITE-KAN framework has been made publicly available at:

https://github.com/N1ne-W/LITE-KAN



(a) LITE-KAN on Car-Hacking.



(b) LITE-KAN on CICIoV2024.

Fig. 5: LITE-KAN Results in the 2-Dimensional Setting.