# KD-GAT: Combining Knowledge Distillation and Graph Attention Transformer for a Controller Area Network Intrusion Detection System

Robert Frenken, Sidra Ghayour Bhatti, Hanqin Zhang, Qadeer Ahmed
The Ohio State University, Columbus, OH, USA
{frenken.2, bhatti.39, zhang.14641, ahmed.358}@osu.edu

***Abstract*—The Controller Area Network (CAN) protocol is widely adopted for in-vehicle communication; however, its lack of inherent security mechanisms makes it vulnerable to various cyber-attacks. This paper presents KD-GAT, an intrusion detection framework based on Graph Attention Networks (GATs) and knowledge distillation (KD), designed to improve detection accuracy while reducing computational complexity. In the proposed approach, CAN traffic is transformed into graph-structured representations using a sliding window technique to capture temporal and relational patterns among messages. A multi-layer GAT with jumping knowledge aggregation serves as the teacher model, and a compact student GAT is trained through a two-phase procedure involving supervised pretraining and knowledge distillation with soft and hard label supervision. Experiments were conducted on three benchmark datasets—Car-Hacking, Car-Survival, and can-train-and-test. Initial results in Car-Hacking and Car-Survival see both the teacher and student perform well, with the student model in particular achieving over achieve 99.97% and 99.31% accuracy, respectively. While train and validation results were promising, the significant class imbalance in the can-train-and-test dataset caused both models to under perform. Future research will need to be conducted to tackle the class imbalance.**

***Index Terms*—Controller Area Network, Intrusion Detection System, Cybersecurity, Graph Attention Network, Knowledge Distillation.**

## I. Introduction

Modern vehicles contain numerous electronic control units (ECUs) that manage nearly all functions of the vehicle, ranging from engine performance to advanced driver assistance systems (ADAS), such as lane departure warning (LDW) and adaptive cruise control (ACC). The controller area network (CAN) protocol, introduced in the mid-1980s for in-vehicle network (IVN), is known for its reliability, cost-effectiveness, and robustness [1]. CAN is widely recognized as the standard communication protocol for ECUs in IVNs. However, the CAN protocol was originally developed with little emphasis on security features such as encryption, message authentication, and access control, as it was assumed to be a closed system without external access points [2].

With the adoption of the on-board diagnostic (OBD) port in all vehicles, direct access to the CAN network was now made possible. Access to the network has only increased with the development of various communication technologies, such as Wi-Fi, vehicle-to-everything (V2X) networks, and cellular networks. This has allowed external sources, including other vehicles and infrastructure, to interact with the vehicle, thus expanding the potential attack surfaces of the vehicle CAN system [3]. Users of open protocols like J1939, understanding this inherent vulnerability, have started developing security solutions, though there are still many cybersecurity gaps to close.

Studies [4]–[6] have shown that cyberattacks on the in-vehicle CAN network can be launched via remote access points (e.g., Wi-Fi, mobile networks, Bluetooth) and physical interfaces (e.g., USB, OBD-II, CD players). Once access is gained, attackers can inject malicious messages to disrupt vehicle functions, potentially taking full control. Attacks on critical systems like brakes or powertrain can lead to severe safety risks. As a result, recent research has focused on enhancing CAN bus security through intrusion detection systems.

With advances in AI, IVN intrusion detection has shifted from monitoring ECU physical characteristics to analyzing CAN data using deep learning and graph-based methods [7], [8], including CNNs and RNNs. However, many supervised models label entire data flows as anomalous, lacking granularity in determining specific malicious messages, which limits detection accuracy and traceability. To address rising CAN network threats, researchers have proposed various intrusion detection systems (IDS), primarily categorized into packet-based and window-based approaches. Packet-based IDSs detect threats quickly using features like ECU clock skew and timing [9], [10], but they can't capture packet sequences, limiting detection of spoofing, flooding, fuzzy, and replay attacks. Window-based IDSs analyze packet correlations, enabling detection and classification. Statistical methods like entropy [11] struggle with low-volume attacks, while survival analysis [12] shows promise but lacks replay attack evaluation. To address limitations in window-based IDSs, a study [7] proposed a statistical method based on graph theory that builds one graph per N packets and uses a chi-square test to identify attack windows. More recent ML-based IDSs, such as the deep convolutional neural network (DCNN)-based IDS [13] and the K-nearest neighbors (KNN)-based IDS [14], aim to reduce such delays while improving detection accuracy.

In [15], an IDS using temporal graph-based features and machine learning (SVM, KNN) is proposed achieving up to

Code available at: https://github.com/RobertFrenken/KD-GAT

97.99% accuracy against DoS, fuzzy, and spoofing attacks. A real-time CAN IDS using dynamic graphs is proposed to detect attacks and identify compromised message IDs efficiently in [16]. It achieves low detection time and memory usage, with theoretical validation via probabilistic analysis. An A&D-GNN-based IDS is proposed in [17] for CAN bus IDS using Arbitration and Data graphs. It achieves 99.92% accuracy on the OTIDS dataset, outperforming existing methods. Finally, KD-XVAE, a lightweight VAE-based IDS uses knowledge distillation and incorporates explainable AI via SHAP for transparent decision-making, ensuring robust and efficient IVN security [18]. ML-based IDSs are hard to adapt and deploy in resource-limited IVNs due to large models and low interpretability. To help address this, techniques such as knowledge distillation and utilizing graph-based features can enable lightweight and adaptable detection.

### A. Motivation

Numerous IDSs have been studied for securing CAN protocol, but each technology has drawbacks. Packet-based IDSs are limited in their ability to analyze the correlation of consecutive packets, and most are unable to classify attacks. Window-based IDS have been studied as well, but development is still in its early stages. To address the limitations of existing IDSs, we propose a graph neural network (GNN)-based teacher model that can learn rich, structural patterns from the CAN traffic. Then, a lightweight student model is trained via knowledge distillation to replicate the teacher's insights—enabling efficient and accurate intrusion detection even on resource-constrained in-vehicle systems.

### B. Contribution

Our main contributions are as follows:

- Creation of a novel Graph Attentional Transformer (GAT) that utilizes jumping knowledge skip connections.
- Employment of a knowledge distillation framework, in response to the resource constrained environments of edge devices.
- Use of multiple benchmark datasets, including the newly created can-train-and-test [19]. To the best of the author's knowledge, this is the first time the whole dataset has been tested with deep learning models.

Figure 1 visualizes the framework. The rest of this paper is organized as follows: Section II gives a comprehensive overview of the literature in this field. Sections III and IV discuss the foundation background of the CAN protocol, Graph Neural Networks, Graph Construction, and Knowledge distillation. Section V discusses the experimental setup and datasets. Section VI shows the results of the experiments. Finally, conclusions are presented in section VII.

## II. Related Work

### A. Literature Review

Several IDS techniques have been explored for securing in-vehicle CAN networks, with common classification based on three key aspects: the number of frames needed for detection, the type of data used, and the detection model design [20]. Based on frame count, IDSs are typically categorized into packet-based and window-based approaches.

Packet-based IDS approaches detect attacks using data from a single CAN packet. Kang et al. [21] employed deep neural networks (NNs) to classify high-dimensional CAN packet features as normal or malicious, though their evaluation was limited to simulation data. Groza et al. [22] developed an IDS leveraging the cyclic nature of in-vehicle traffic and fixed data field formats, using Bloom filters to assess frame periodicity. However, this method is only effective for periodic frames, not aperiodic ones [23]. While packet-based IDSs offer fast detection, they lack the ability to analyze correlations between consecutive packets, limiting their accuracy.

Window-based IDSs analyze CAN packet sequences within fixed sizes or time frames. Olufowobi et al. proposed a real-time IDS using timing models based on message intervals and worst-case response times. It detects anomalies without predefined specs but struggles with aperiodic messages and repeated IDs [24]. Several window-based IDSs use frequency analysis. Taylor et al. analyzed CAN packet frequency and Hamming distance [25], but such methods struggle with aperiodic packet attacks [26], [27]. Islam et al. used graph features from fixed-size windows with chi-squared and median tests to detect anomalies and replay attacks, but their approach needs many packets, reducing responsiveness [7]. Graph-based IVN IDSs model ECU interactions well but mainly detect simple attacks G-IDCS addresses this by combining a threshold-based (TH) classifier for reduced window size and interpretability with an ML classifier that uses message correlations to classify attack types—something packet-based IDSs miss [28].

## III. Background

### A. CAN Protocol

In this section, we will provide an overview of the CAN protocol and graph neural networks (GNNs).

The CAN is a serial communication system designed to efficiently handle distributed real-time control, and it is widely used for communication between electronic control units (ECUs) in vehicles. In a CAN bus setup, transmitting nodes broadcast messages, while receiving nodes use filtering mechanisms to decide whether to process the messages. As illustrated in Figure 2, a CAN data frame is made up of seven fields. First is start-of-frame (SoF), a single dominant bit, then arbitration field, an 11-bit identifier and a 1-bit RTR. Next is the control field, a 1-bit IDE, a reserved bit, and a 4-bit DLC that specifies the data field's length. The data field varies in size according to the DLC and holds the actual payload. The CRC field features a 15-bit cyclic redundancy check and a 1-bit delimiter with a recessive bit. The ACK field includes a dominant bit for acknowledgment and a recessive delimiter. Finally, the EoF field ends the frame with seven recessive bits.

### B. Graph Neural Networks

A graph is a data structure consisting of a set of nodes $V$ and a set of edges $E$ that connect pairs of nodes. A graph can be

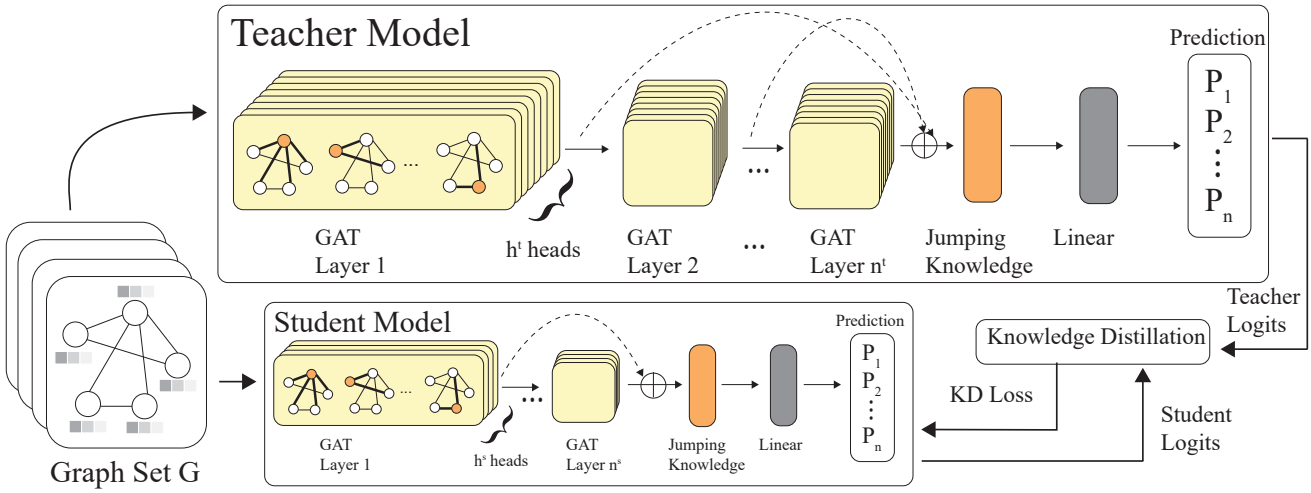


Fig. 1. KD-GAT knowledge distillation framework. Variables $h$ and $n$ denote the number of attention heads and number of layers, while $t$ (teacher) and $s$ (student) denotes the hyperparameters for said variables.

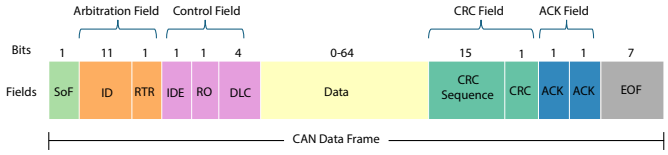


Fig. 2. CAN data frame structure

defined as $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is a node set with $n$ nodes, and $E = \{e_1, e_2, ..., e_m\}$ is an edge set with $m$ edges. Given this graph structure, a graph neural network (GNN) looks to find meaningful relationships and insights of the graph. The most common way to accomplish this is through the message passing framework [29] [30], where at each iteration, every node aggregates information from its local neighborhood. Across iterations, node embeddings contain information from further parts of the graph. This update rule can be explained through the following equation:

$$\mathbf{h}_v^{(k)} = \phi\left(\mathbf{h}_v^{(k-1)}, \bigoplus_{u \in \mathcal{N}(v)} \psi\left(\mathbf{h}_v^{(k-1)}, \mathbf{h}_u^{(k-1)}, \mathbf{e}_{vu}\right)\right) \quad (1)$$

where $\phi$ is the update function, $\psi$ is the message function, $\bigoplus$ is the aggregation operator (e.g., sum, mean), and $\mathcal{N}(v)$ is the neighbors of node $v$.

GAT [31] builds upon GNNs by introducing an attention mechanism. This allows each node in the message passing framework to dynamically assign weight contributions to their neighbors. For node $v$, the attention coefficient $\alpha_{vu}$ for neighbor $u$ is computed as:

$$\alpha_{vu} = \text{softmax}\left(\text{LeakyReLU}\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_v \parallel \mathbf{W}\mathbf{h}_u\right]\right)\right) \quad (2)$$

where $\alpha$ is the learnable attention vector, $\mathbf{W}$ is a weight matrix, and $||$ denotes concatenation between the weight matrices.

The attention function computes a scalar weight for each neighbor of node $v_i$, denoted by $\alpha_{ij}$, which reflects the importance or relevance of node $v_j$ for node $v_i$.

$$\mathbf{h}_v^{(k)} = \sigma\left(\sum_{u \in \mathcal{N}(v)} \alpha_{vu} \mathbf{W}\mathbf{h}_u^{(k-1)}\right) \quad (3)$$

where $\sigma$ is the activation function, normally ELU or RELU.

The Jumping Knowledge (JK) module [32] further enhances GATs by combining intermediate layer representations, similar to a skip connection. This paper uses the LSTM-based aggregation, where $h_v^{(l)}$ denotes node $v$'s representation at layer $l \in 1, ..., L$. The JK module gathers all layer outputs, performs a forward LSTM pass through layers (1→L), and a backward LSTM pass through the same layers in reverse (L→1), where they are concatenated for a final representation. Equations 4, 5, and 6 demonstrate this process.

$$\overrightarrow{h}_v^{(l)} = \text{LSTM}\left(h_v^{(l)}, \overrightarrow{h}_v^{(l-1)}\right) \quad (4)$$

$$\overleftarrow{h}_v^{(l)} = \text{LSTM}\left(h_v^{(l)}, \overleftarrow{h}_v^{(l+1)}\right) \quad (5)$$

$$h_v^{\text{final}} = \left[\overrightarrow{h}_v^{(l)}; \overleftarrow{h}_v^{(l)}\right] \quad (6)$$

## IV. Methodology

In this section, the construction of graph and knowledge distillation is explained in detail.

### A. Graph Construction

CAN data is normally in the form of a tabular dataset, where each row denotes a message with attributes such as the CAN ID, payload and timestamp. A natural question then is: how can time series data be turned into a graph? First, due to the low latency of the CAN bus (a maximum of approximately 8,771 messages per second), ECU's can rapidly respond to each other. It is implied that Message A at time $t$, and Message B at time $t + 1$ are a call and response to each other. Given

this relational significance, an edge can be created between message A and B. Second, it is understood that a collection of messages close together represents the vehicle's state in the window of time. This "window" can be used to determine the size of the graph. This paper will use a window size of 50 messages, striking a good balance of enough collected messages, while limiting latency for real time prediction. Following [17], given a window size, a graph dataset is created where each node has the attributes of CAN ID, CAN ID frequency in window, and average data field value. The edge list takes pairs of nodes that are together sequentially, and the number of occurrences of said edge. Figure 3 shows some examples of attack-free and attack graphs.

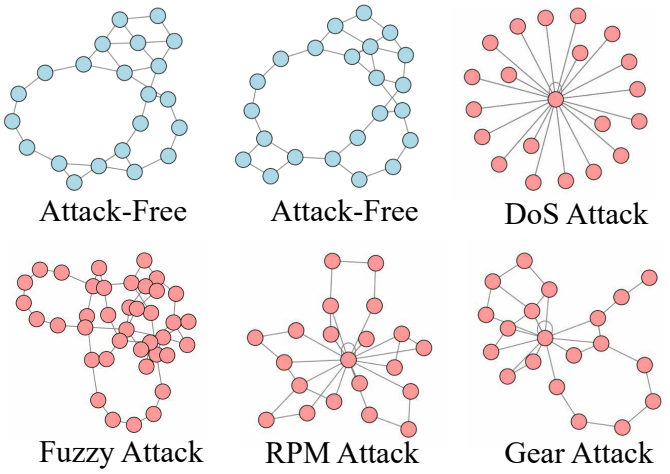


Fig. 3. Graphs created from the Car Hacking Dataset. Each node represents a unique CAN ID found in the window, and each edge is constructed between sequential IDs, including self edges. Not shown is the feature vector associated with each node. Blue denotes that the graph has no attack IDs, while red indicates that at least one of the nodes is an intrusion.

### B. Knowledge Distillation

Knowledge distillation (KD), popularized by Hinton et al. [33], is a widely adopted model compression technique where a small, efficient student model is trained to reproduce the behavior of a large, accurate teacher model. The soft target probabilities output by a teacher model encode rich relational information between classes that's often not captured by hard labels alone. Training a student model to match these softened outputs enables it to learn a more informative function approximation than training with one-hot labels alone.

Concretely, given an input $x$, the teacher produces a vector of logits $s^t(x)$, which are converted into a softened distribution via temperature scaling $\tau$:

$$\tilde{p}_k^t(x) = \frac{\exp(s_k^t(x)/\tau)}{\sum_j \exp(s_j^t(x)/\tau)} \tag{7}$$

The student is trained to match these probabilities by minimizing the Kullback-Leibler divergence between teacher and student distributions (distillation loss), alongside the standard supervised classification loss:

$$\mathcal{L}_{\text{total}} = \alpha \cdot \mathcal{L}_{\text{hard}} + (1-\alpha) \cdot \mathcal{L}_{\text{KD}} \tag{8}$$

where $\alpha$ balances the contribution of ground truth and teacher supervision.

A possible real world implementation would be for the teacher and student to stay on a cloud server, learning from received data, while a copy student model is onboard the vehicle making real time inferences. Once the student has marginally improved, periodic weight updates could be sent from the server to the student models on the vehicles, similar to the occasional phone update users get on their edge devices.

## V. Experiments

### A. Experimental Setup

All experiments were conducted using PyTorch and PyTorch Geometric. Model training and evaluation were performed on GPU clusters provided by the Ohio Supercomputer Center (OSC). Table I highlights the differences between the teacher and student model. The KD framework operates where the teacher network trains first, followed by the student network which consists of two stages:

- **Stage 1:** Pretrain the student model using binary cross-entropy (BCE) loss only, known as a "warm up" period.
- **Stage 2:** Fine-tune the student using a mixed loss, combining BCE with a distillation loss using logits from the teacher and student.

Table II details the hyper-parameters for this experiment. Finally, in each experiment, 80% of the dataset was utilized for training, 20% for validation, and a distinct test set was used compiled by the dataset providers.

TABLE I
Teacher and Student Model Parameters

| Parameter | Teacher | Student |
|---|---|---|
| GAT layers | 5 | 2 |
| Attention heads | 8 | 4 |
| Hidden channels | 32 | 32 |
| Linear Layers | 3 | 3 |
| Loss function | BCE | KL Divergence |
| Total Parameters | 4,999,426 | 316,034 |

TABLE II
Hyperparameters

| Hyperparameter | Value | Notes |
|---|---|---|
| Learning rate | 0.0005 | Adam optimizer |
| Batch size | 128 | - |
| Epochs | 100 | Saved best validation epoch |
| Student warm-up epochs | 5 | Distillation only after warm-up |
| Dropout | 0.2 | Across all GAT and linear layers |
| Temperature scaling | 2.0 | For soft targets |
| $\alpha$ | 0.5 | Distillation coefficient |
| Window size | 50 | Sliding window length |
| Stride size | 50 | Step between windows |

### B. Datasets

Our proposed method has been evaluated over three datasets: Car-Hacking, Car-Survival, and can-train-and-test. [1]

[1]https://bitbucket.org/brooke-lampe/can-train-and-test-v1.5/src/master/

- **HCRL Car-Hacking:** Real data from a Hyundai Sonata. Attacks include DoS, fuzzy, and spoofing (RPM and drive gear). [34]
- **HCRL Survival Analysis:** Real data from a Chevrolet Spark, Hyundai Sonata, and Kia Soul. Attacks include DoS, fuzzy, and spoofing. [35]
- **can-train-and-test:** Real data from a 2011 Chevrolet Impala, 2011 Chevrolet Traverse, 2016 Chevrolet Silverado, and 2017 Subaru Forester. Attack scenarios include DoS, fuzzing, gear spoofing, speed spoofing, interval, standstill, systematic, and combined spoofing. Each of these vehicles are trained and tested in their own unique set, labeled in this paper from set_01 to set_04. [19]

One important note is that the dataset can-train-and-test has extreme class imbalance, where the ratio of attack-free to attack samples range from 36:1 up to 927:1 depending on the subset. This class imbalance will be reflective in the results section, and will tie into a discussion of future work. As an initial measure, the loss functions of both models were put through an additional function called focal loss, which aims to focus learning on hard misclassified examples. Below is the equation, where $\gamma$ is the focusing parameter. Our experiment used a moderate $\gamma$ of 1.0.

$$FL(p_t) = -(1-p_t)^{\gamma}\ \log(p_t) \tag{9}$$

### C. Evaluation Metrics

We report accuracy, precision, recall, and F1-score, defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Precision = \frac{TP}{TP + FP} \tag{11}$$

$$Recall = \frac{TP}{TP + FN} \tag{12}$$

$$F1 - Score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

where TP, TN, FP, and FN denote true/false positives/negatives.

## VI. Results and Discussion

### A. Results

Below are the results of the experiments described in previous sections. The comparison models used were A&D GAT [17] and ECF-IDS [36], an IDS utilizing a cuckoo filter and a BERT model. These were picked as one is the only know GAT model for CAN IDS, while the other is the only known DL model that has evaluated the can-train-and-test dataset. The performance of the model was assessed using the validation set and held out test set provided by [19].

TABLE III
Validation and Test performance for all methods on CAN intrusion datasets.

| **Validation** | | | | | |
|---|---|---|---|---|---|
| **Dataset** | **Method** | **Acc.** | **Prec.** | **Recall** | **F1** |
| Car Hacking | A&D [17] | 0.9995 | 0.9994 | 0.9993 | 0.9994 |
| | Teacher | 0.9995 | 1.0000 | 0.9983 | 0.9991 |
| | Student-KD | 0.9995 | 1.0000 | 0.9982 | 0.9991 |
| Car Survival | Teacher | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| | Student-KD | 1.0000 | 1.0000 | 1.0000 | 1.0000 |
| Set_01 | Teacher | 0.9981 | 1.0000 | 0.8908 | 0.9423 |
| | Student-KD | 0.9991 | 1.0000 | 0.9478 | 0.9732 |
| Set_02 | Teacher | 0.9740 | 0.9980 | 0.6983 | 0.8216 |
| | Student-KD | 0.9959 | 0.9970 | 0.9553 | 0.9757 |
| Set_03 | ECF-IDS [36] | 0.9996 | 0.9991 | 0.9984 | 0.9986 |
| | Teacher | 0.9963 | 1.0000 | 0.8897 | 0.9416 |
| | Student-KD | 0.9977 | 0.9976 | 0.9335 | 0.9645 |
| Set_04 | Teacher | 0.9977 | 1.0000 | 0.8509 | 0.9194 |
| | Student-KD | 0.9975 | 1.0000 | 0.8362 | 0.9108 |
| **Test** | | | | | |
| **Dataset** | **Method** | **Acc.** | **Prec.** | **Recall** | **F1** |
| Car Hacking | Teacher | 0.9977 | 0.9969 | 0.9985 | 0.9977 |
| | Student-KD | 0.9997 | 1.0000 | 0.9993 | 0.9997 |
| Car Survival | Teacher | 0.9695 | 0.9415 | 0.9987 | 0.9692 |
| | Student-KD | 0.9931 | 0.9897 | 0.9961 | 0.9929 |
| Set_01 | Teacher | 0.9923 | 1.0000 | 0.7706 | 0.8705 |
| | Student-KD | 0.9929 | 0.9993 | 0.7874 | 0.8808 |
| Set_02 | Teacher | 0.9908 | 0.5446 | 0.2473 | 0.3402 |
| | Student-KD | 0.9818 | 0.2034 | 0.3055 | 0.2442 |
| Set_03 | Teacher | 0.9698 | 1.0000 | 0.5811 | 0.7350 |
| | Student-KD | 0.9824 | 1.0000 | 0.7554 | 0.8606 |
| Set_04 | Teacher | 0.8831 | 0.9999 | 0.4960 | 0.6631 |
| | Student-KD | 0.8707 | 1.0000 | 0.4425 | 0.6135 |

### B. Discussion

Looking at the results, both the teacher and student perform as well or better with comparable models for the Car Hacking and Car Survival datasets in both the validation and test sets. The results from can-train-and-test are also promising with the validation set. However, when testing on unseen test data, the large discrepancy of class ratios cause metrics such as precision, recall, and F1-score to under perform. The only paper outside the original authors who used dataset [19] was [36] with their model ECF-IDS. However, they only provide results for Set_03, so it's unclear how other deep learning models perform across all subsets of can-train-and-test. Measures such as different dropout rates, class weight balancing, and focal loss were tried to alleviate the class imbalance, but these experiments were not able to perform significantly better.

## VII. Conclusion

In this study, we propose an approach that takes CAN data and turns it into graph data. With this graph data, we combine the concepts of a graph attention network and knowledge distillation to utilize the graph properties of the dataset, and to create an accurate model that aims to meet the size requirements for an edge device. Experimentation was conducted using three benchmark datasets: Car-Hacking, Car-Survival, and can-train-and-test. On Car-Hacking and Car-survival, both the teacher and student performed well, with the student achieving 99.97% and 99.31% accuracy on the respective datasets. Though the validation metrics were high on can-train-and-test, upon further inspection the severe class imbalance caused the testing metrics such as precision, recall, and F1 to under perform. Future research will need to be conducted to find root problems and solutions to the class imbalance, as this imbalance will hold for applications with real world CAN data.

## References


[1] K. Pazul, "Controller area network (can) basics," Preliminary DS00713A, AN713, Microchip Technology Inc., 1999.

[2] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Lowlevel communication characteristics for automotive intrusion detection system," *IEEE Transactions on Information Forensics and Security*, vol. 13, pp. 2114–2129, Aug. 2018.

[3] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," in *Black Hat USA*, vol. 2015, p. 91, Aug. 2015.

[4] S. Woo, H. J. Jo, and D. H. Lee, "A practical wireless attack on the connected car and security protocol for in-vehicle can," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 993–1006, Sep. 2015.

[5] Y. Lee, S. Woo, J. Lee, Y. Song, H. Moon, and D. H. Lee, "Enhanced android app-repackaging attack on in-vehicle network," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–13, Feb. 2019.

[6] H. Wen, Q. A. Chen, and Z. Lin, "Plug-n-pwned: Comprehensive vulnerability analysis of obd-ii dongles as a new over-the-air attack surface in automotive iot," in *Proceedings of the 29th USENIX Security Symposium*, pp. 949–965, 2020.

[7] R. Islam, R. U. D. Refat, S. M. Yerram, and H. Malik, "Graph based intrusion detection system for controller area networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 1727–1736, Mar. 2022.

[8] S. B. Park, H. J. Jo, and D. H. Lee, "G-idcs: Graph-based intrusion detection and classification system for can protocol," *IEEE Access*, vol. 11, pp. 39213–39227, 2023.

[9] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proceedings of the 25th USENIX Security Symposium*, pp. 911–927, 2016.

[10] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of can messages for in-vehicle network," in *Proceedings of the International Conference on Information Networking (ICOIN)*, pp. 63–68, Jan. 2016.

[11] M. Muter and N. Asaj, "Entropy-based anomaly detection for in-vehicle networks," in *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pp. 1110–1115, Jun. 2011.

[12] M. L. Han, B. I. Kwak, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, Oct. 2018.

[13] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, Jan. 2020. Art. no. 100198.

[14] A. Derhab, M. Belaoued, I. Mohiuddin, F. Kurniawan, and M. K. Khan, "Histogram-based intrusion detection and filtering framework for secure and safe in-vehicle networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 2366–2379, Mar. 2022.

[15] R. U. D. Refat *et al.*, "Detecting can bus intrusion by applying machine learning method to graph based features," in *Intelligent Systems and Applications: Proceedings of the 2021 Intelligent Systems Conference (IntelliSys) Volume 3*, Springer International Publishing, 2022.

[16] J. Song *et al.*, "Dgids: Dynamic graph-based intrusion detection system for can," *Computers & Security*, vol. 147, p. 104076, 2024.

[17] Y. He *et al.*, "A&d graph-based graph neural network intrusion detection for in-vehicle controller area network," in *2024 IEEE/CIC International Conference on Communications in China (ICCC)*, IEEE, 2024.

[18] M. A. Yagiz *et al.*, "Transforming in-vehicle network intrusion detection: Vae-based knowledge distillation meets explainable ai," in *Proceedings of the Sixth Workshop on CPS & IoT Security and Privacy*, 2024.

[19] B. Lampe and W. Meng, "can-train-and-test: A curated can dataset for automotive intrusion detection," *Computers & Security*, vol. 140, p. 103466, 2024.

[20] G. Dupont, J. D. Hartog, S. Etalle, and A. Lekidis, "A survey of network intrusion detection systems for controller area network," in *Proceedings of the IEEE International Conference on Vehicle Electronics and Safety (ICVES)*, pp. 1–6, Sep. 2019.

[21] M.-J. Kang and J.-W. Kang, "A novel intrusion detection method using deep neural network for in-vehicle network security," in *Proceedings of the IEEE 83rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5, May 2016.

[22] B. Groza and P.-S. Murvay, "Efficient intrusion detection with bloom filtering in controller area networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, pp. 1037–1051, Apr. 2019.

[23] Y. Wei, C. Cheng, and G. Xie, "Ofids: Online learning-enabled and fingerprint-based intrusion detection system in controller area networks," *IEEE Transactions on Dependable and Secure Computing*, Dec. 19 2022. early access.

[24] H. Olufowobi, C. Young, J. Zambreno, and G. Bloom, "Saiducant: Specification-based automotive intrusion detection using controller area network (can) timing," *IEEE Transactions on Vehicular Technology*, vol. 69, pp. 1484–1494, Feb. 2020.

[25] A. Taylor, N. Japkowicz, and S. Leblanc, "Frequency-based anomaly detection for the automotive can bus," in *Proceedings of the World Congress on Industrial Control Systems Security (WCICSS)*, pp. 45–49, Dec. 2015.

[26] M. Bozdal, M. Samie, and I. K. Jennions, "Winds: A wavelet-based intrusion detection system for controller area network (can)," *IEEE Access*, vol. 9, pp. 58621–58633, 2021.

[27] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, pp. 6123–6141, Jul. 2022.

[28] S. B. Park, H. J. Jo, and D. H. Lee, "G-idcs: Graph-based intrusion detection and classification system for can protocol," *IEEE Access*, vol. 11, pp. 39213–39227, 2023.

[29] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1263–1272, 2017.

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[31] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph attention networks," in *International Conference on Learning Representations (ICLR)*, 2018. Available: https://arxiv.org/abs/1710.10903.

[32] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-i. Kawarabayashi, and S. Jegelka, "Representation learning on graphs with jumping knowledge networks," *arXiv preprint arXiv:1806.03536*, 2018.

[33] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[34] H. M. Song, J. Woo, and H. K. Kim, "In-vehicle network intrusion detection using deep convolutional neural network," *Vehicular Communications*, vol. 21, p. 100198, 2020.

[35] S. Han, H. K. Kim, and H. K. Kim, "Anomaly intrusion detection method for vehicular networks based on survival analysis," *Vehicular Communications*, vol. 14, pp. 52–63, 2018.

[36] S. Li, Y. Cao, H. J. Hadi, F. Hao, F. B. Hussain, and L. Chen, "Ecf-ids: An enhanced cuckoo filter-based intrusion detection system for in-vehicle network," *IEEE Transactions on Network and Service Management*, 2024. Early Access.