# The Effectiveness of Using GNN for Detection of DoS Attacks in the Security Message Transmission System in VANETs

Samuel Henrique Miranda Alves , Aldri Luiz dos Santos[1]

[1]Departamento de Ciência da Computação (DCC)
Instituto de Ciências Exatas (ICEx)
Universidade Federal de Minas Gerais (UFMG)
January 2025

{samuelhenrique,aldri}@dcc.ufmg.br

***Abstract.*** *Denial of Service attacks have become a disrupting issue for modern applications. In the context of autonomous vehicles, this problem can affect the users in many ways, ranging from security data breaches to a crash in the car's system, which prevents the broad availability of these services to society. In recent years, a vast majority of studies have proposed the use of contemporary Machine Learning techniques to assist in the detection of these anomalies. Nevertheless, they still cannot cope with the fast-paced nature of this adversarial attack. In this work, we propose to study a trendy method called Graph Neural Networks to evaluate its effectiveness over these challenges in the vehicular network context, specifically regarding the security message transmission system, using a publicly available dataset. The measured metrics achieved great performance compared to other traditional classifiers, which emphasizes the robustness of this model and paves the way for future works looking to assess stronger variables and sophisticated scenarios.*

## 1. Introduction

Given the quick development of network systems in the past few years, the technologies concerning the scenario of Intelligent Transportation Systems (ITS) have drawn a lot of attention, because they provide smart solutions for the problems related to vehicle traffic [Alam et al. 2016]. Among the different advantages, we can state the smart monitoring and control of land transportation, improvement of the security service quality, deployment of mobility services guided to users, etc. All of this contributes to the increase in the efficiency of transport and the prevention of dangerous scenarios, such as accidents [Hadded et al. 2015]. Yet, despite all the tests and investments being conducted in this area, there is still a significant number of obstacles that impede the delivery of this technology to society.

Autonomous vehicular network systems operate based on the exchange of safety messages, known as Basic Safety Messages (BSM). They can be used by safety applications to prevent hazardous situations in time. In the face of the heterogeneous context and ephemeral characteristics of a vehicular network, these messages must be exchanged rapidly between network devices (on average, 10 times per second or every 100 ms) to enable continuous monitoring of the system's elements. This necessity to rapidly share data with multiple neighboring vehicles, infrastructure units, and other devices increases the

system's vulnerability to potential intruder attacks. Among the areas where such attacks cause damages, the system's availability arises as one of the most critical security functionalities, as this directly impacts communication between network components. In this context, Denial of Service attacks (DoS) represent one of the most well-known and recurring types of intrusion. DoS attacks occur when a malicious agent disrupts services by interfering with the channel, overloading the network, and/or causing packet loss, rendering the network fully or partially unavailable to genuine users [Ahmed and Elhadef 2019]. This involves one of the most common attacks in a vehicular network and can be perpetrated by an attacker both within and outside the network.

Following these lines, a lot of research over the years has investigated the use of Machine Learning methods to assist in the protection of these systems. Even though they achieved great accuracy, most of the times they were not considered for use in the real-world application scenarios [Sommer and Paxson 2010]. This can be explained because the studied models underlie their analysis from the network traffic data to find features that allow the recognition of the attacks. Considering the context of computer networks, the applications in real life hold a dynamic configuration, which means that these features can be easily modified for attackers to conceal their identity [Corona et al. 2013]. Thereby, new solutions focused on studying the attack structure pattern are required, instead of working on the features of the data flow, in order to make the methods more robust. We could achieve this by modeling the network as a graph, as this allows visualization of the arrangement of the elements and, thus, by applying algorithms on the graph topology, we are able to examine the structure of the network and the relationship of its components. Unlike the features from the traffic data, the attacks' intrinsic patterns cannot be modified by an intruder, which is a better way to tackle this issue.

Therefore, in this work, we intend to analyze the effectiveness of using Graph Neural Networks (GNN) to assist in the detection of DoS attacks for the security message transmission system of Vehicular Ad-hoc Networks. The idea consists of shaping the network elements in a graph form and classifying its feature patterns into a benign or malignant flow, to capture not only the characteristics of individual components, but also their relationships within the network. For the experimental part of this project, we first chose a publicly available dataset named DARE [HAIDAR et al. 2020]. This dataset contains reports of anomalous behaviors entities identified by the vehicles during the exchange of BSM messages, as well as the content of each message. We, then, filtered the data to consider only the DoS type of attacks and used it to train the algorithm deployed in our previous work.

In the next phase, we focused on selecting metrics to evaluate our proposal. We initially considered metrics that measure the performance of the algorithm and its efficiency for the classification task. After that, we also tested the same dataset with different models of Machine Learning to compare the results yielded between them. Finally, we selected some explainable graph metrics to assess the robustness of our model to remain solid over potential changes in the network traffic. The final results showed that our solution outperformed all the other classifiers used for comparison, besides keeping the same accuracy when artificial modifications were applied to the dataset, whereas the other models decreased their respective metrics. Hence, our work sheds light on the potential of using Graph Neural Networks for the job of detecting intrusion anomalies in the dynamic

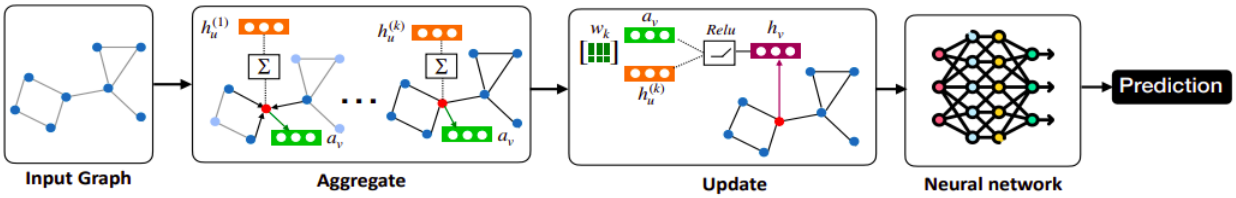


**Figure 1. Training steps of a GNN. [MITRA et al. 2024]**

environment of vehicle networks.

## 2. Background

In our project, we are broadly interested in applying GNN methods to support the cybersecurity systems of vehicular communications. In the following subsections, we will introduce the main concepts regarding these topics and review the current state-of-the-art, based on recent studies that cover the proposed areas of this work.

### 2.1. Graph Neural Networks

Graph Neural Networks (GNNs) are a recent neural network family in the Artificial Intelligence scope specifically designed to learn and generalize over graph-structured data, by capturing and modeling the inherent patterns in graphs [Scarselli et al. 2008]. The essential foundation of GNNs relies on converting the graph structure of our problem into a valid input for classical Neural Network (NN) frameworks to perform traditional machine learning tasks, such as classification, regression, or clustering. The representation of the problem data using a graph form emerges as a good way to provide the means of capturing complex relationships between entities, utilizing the relations with the two basic components: nodes and edges [Franceschi et al. 2019].

There are mainly three general problems for graphs in the machine learning scope: node classification, link prediction, and graph classification [Wu et al. 2023]. In node-level classification, the GNN model assigns each node to one or more predefined classes or predicts a continuous value associated with the node based on its node feature embedding from the neighboring nodes. Link prediction is the problem of predicting new links between the nodes [Xian et al. 2022]. In graph-level classification, the GNN should learn to classify entire graphs into specific categories. The GNN should effectively capture and aggregate information from all the nodes and edges in the graph to make predictions about the graph as a whole.

The GNN ability to collect the inherent data from the underlying complex relationships of a graph stems from two prime functions of the GNN framework: (1) aggregation and (2) update, as shown in Figure 1. Each node has its own vector of embedding features, which can also include information about its edges. Starting from an initial part of the graph and repeating this process over multiple iterations, the framework learns the information of one specific node from its vector features and shares this data with its neighbors. The received information is then aggregated to form the input for the next step. This is called the aggregation process. The update step simply consists of updating the nodes' embedding using the learned/aggregated information [Xu et al. 2019]. Since the emergence of GNN frameworks in the last decade [Scarselli et al. 2008], many GNN

methods have been proposed, which only differentiate from the various AGGREGATE(·) and UPDATE(·) functions created, such as Graph Convolutional Networks (GCN) and Graph Attention Network (GAT).

One of the popular approaches to learning with graph-structured data is to make use of graph kernels (functions that measure the similarity between graphs) plugged into a kernel machine, such as a support vector machine [Kriege1 et al. 2020]. Kernel methods refer to machine learning algorithms that learn by comparing pairs of data points using particular similarity measures — kernels. Another popular approach to training unsupervised GNNs is through the use of classic autoencoder frameworks. Autoencoders were first introduced in [McClelland and Rumelhart 1986] as neural networks that are trained to reconstruct their inputs. Specifically, an autoencoder consists of two components, an encoder and a decoder. The encoder compresses each data point into a low-dimensional vector representation, whereas the decoder works to reconstruct the original information from that vector.

However, existing graph autoencoders lack the motivation to represent an entire neighborhood of the graph nodes, and are primarily designed to decode only direct links between pairs of nodes, resulting in a minimization of the link reconstruction loss. The fundamental difficulty in reconstructing all receptive fields of GNNs is due to the non-trivial design of a reconstruction loss in the irregular structures of the graph. Unfortunately, oversimplification in connection reconstruction causes the learned node representations to lose a lot of information and thus provides undesirable performance in many downstream tasks, which is one of the challenges for the use of GNNs, especially for wide-data problems that generate sparse and dense graphs.

### 2.2. Vehicular Ad-Hoc Networks

Vehicular Ad-Hoc Networks (VANETs) are a complex vehicle communication system based on smart vehicles and base stations that share information via wireless communications. It can also include other communication entities around them, such as roadside units, clouds, fog and grid networks, and Internet devices carried by individuals and pedestrians, which creates a dynamic and challenging network. Automated Vehicles (AV) are categorized into six levels of automation, ranging from 0 to 5, as shown in Figure 2 [Lamnabhi-Lagarrigue et al. 2017]. We are currently at level 2, which contains a partial degree of automation. Vehicles above level 3 are still under research and do not require the driver to keep their hands on the steering wheel. The evolution of Connected Autonomous Vehicles (CAVs) towards the full level of driving automation indicates that futuristic vehicles will be very dependent on sensors and that navigation decisions will be dependent on the quality of the collected data.

Autonomous vehicular network systems operate based on a cooperation mechanism between stations located both within the road infrastructure and in the components of the vehicles themselves (Intelligent Transportation Systems Stations - ITS-S). These stations, such as the On-Board Units (OBUs) in vehicles or the Road-Side Units (RSUs) on the roadways, send and receive messages throughout the vehicular network [Lu et al. 2014]. Vehicles can exchange messages with each other, a type of architecture known as V2V, or with the network infrastructure, known as V2I communication, as illustrated in Figure 3. The safety messages exchanged between them can be transmitted via full broadcast, sent regularly throughout the network, and contain information such as the

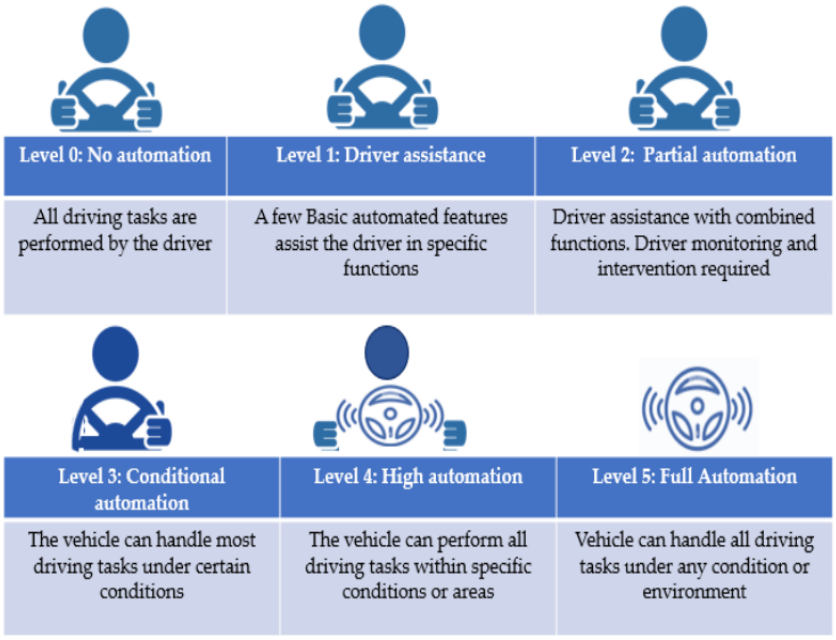


**Figure 2. Driving automation levels. [Baccari et al. 2024]**

position, speed, and acceleration of nearby vehicles or event-specific alert messages, such as roadworks or emergency stops [Nagarajan et al. 2023]. These safety messages, known as Basic Safety Messages (BSM), are used by safety applications to prevent hazardous situations in time.

Due to sensor data uncertainties caused by cyber attacks, autonomous driving systems require a sophisticated design to capture those abnormalities and eventually mitigate their impacts [Wang et al. 2020]. DoS attacks act as one of the most common types of intrusions, as they can be perpetuated in different ways. There are two main possible implementations of DoS attacks in the context of VANETs: (1) the attacker repeatedly sends false accident messages to a nearby vehicle, keeping it occupied with the process of verifying the message's authenticity instead of focusing on useful communication processes; and (2) the attacker targets RSUs with the same strategy, making the communication system with other vehicles unavailable. Therefore, a solution aimed at mitigating this type of attack must consider these two possible scenarios. Because of this, anomaly detection is an essential task to guarantee the safety of autonomous vehicles and the certainty of their decisions.

Thus, it is mandatory to implement Anomaly Detection Systems (ADS) capable of mitigating the negative impacts that anomalies can cause on the navigation decisions of the CAVs [Masmoudi et al. 2019]. An ADS for CAVs is a collection of mechanisms/algorithms that makes it possible to identify, isolate, and prevent any deviation from the normal state of the CAV system towards an abnormal situation due to several causes discussed previously. The ADS is characterized by several tasks, primarily monitoring the system state and collecting data against anomalies using advanced algorithms. Evaluation metrics help improve the reliability and safety of driverless vehicles by ensuring high-quality solutions for preventing potential incidents and detecting anomalies. Several

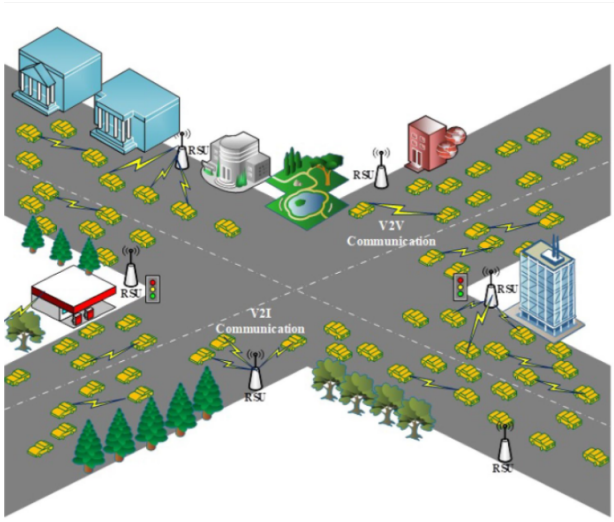


**Figure 3. Communication types in a VANET [Nagarajan et al. 2023]**

metrics are used to measure the technique's performance and they are varied depending on the type of approach used (i.e., Machine Learning, Deep Learning, or Statistical Learning). These metrics mainly include Accuracy, Precision, Recall, F1-score, Mean Squared Error (MSE) and more [Limbasiya et al. 2022], which are some of the evaluations we used in the experimental part of this work.

## 3. Related works

Despite being a topic that has only recently sparked interest in the scientific community, there are already a variety of studies addressing the two main subjects of this work, which are the use of GNNs in the context of cyber-attacks and the security of autonomous vehicle systems. Among them, the main reference to be followed is [da Silva et al. 2023]. In their work, the authors proposed the creation of a learning methodology in VANETs topology that prioritizes data anonymization and can be used with any graph learning method. Additionally, the study confirms the quality of using graph models applied within the context of vehicular networks and autonomous cars. Thus, the authors conclude that even with small pieces of information regarding graph topology, in order to respect user privacy, it is possible to extract important and relevant information that can be useful in traffic analysis and accident prevention in the era of autonomous vehicles.

In this other study [Pujol-Perich et al. 2021b], which will also serve as a guide for this research, the authors explore a different way of structuring the network as a graph, referred to as a host connection graph. They propose representing each network flow as a node in the graph. Given a set of flows, a host-connection graph includes a node for each distinct host involved—whether sending or receiving traffic. Thus, for a flow f, with a source host S and a destination host D, two undirected edges are created: one from the source host to the flow ($S \rightarrow f$) and another from the flow node to the destination host ($f \rightarrow D$). A simpler representation would consider only the hosts as the graph nodes and the flows as edges connecting the source and destination hosts. However, the decision to add specific nodes representing each flow was motivated by the operational principles of GNN models. GNNs treat the hidden states of nodes in input graphs as the only learnable objects. Consequently, to properly learn the characteristics of flows, it is necessary to

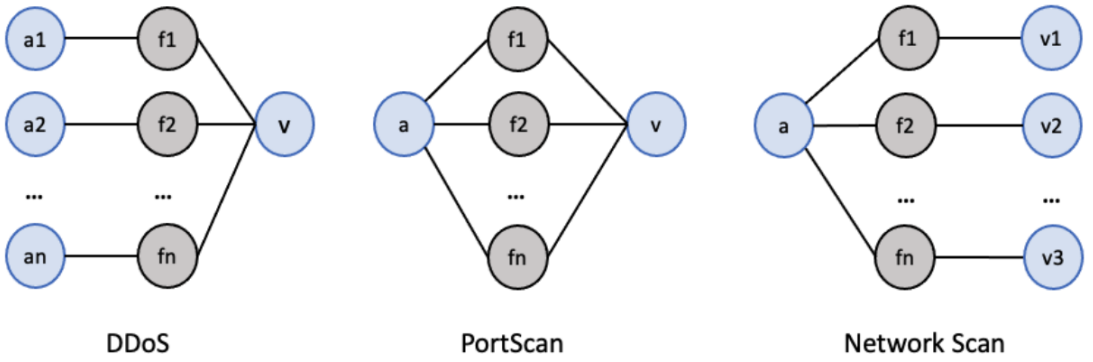


**Figure 4. Cyber-attacks flows represented as graphs. [Pujol-Perich et al. 2021a]**

add them as nodes in the graph. This approach is particularly interesting for learning about network flows in the context of cyberattacks, as attacks follow unique patterns that can be identified through their inherent behaviors. The research results reveal significant performance consistency when presenting the proposed model with unseen data during training, a characteristic that is challenging for more traditional models to achieve. A graphical representation of the concept discussed in this paper can be seen in Figure 4.

Similarly, within the approach of Machine Learning methods for intrusion detection in connected autonomous vehicle environments, a study by [Nagarajan et al. 2023] analyzes the current state-of-the-art in this field, providing extensive coverage of the various areas that need to be examined to achieve satisfactory results. The authors aim to present a comprehensive discussion of the existing concepts in the field of autonomous vehicles, addressing the various types of systems and communications within the network, and highlighting the current strategies being employed in the field of Machine Learning to address cybersecurity issues within each of these domains. Among the areas discussed, they examine the operation of intra-vehicular communications (communication between devices within the vehicle) and inter-vehicular communications (communication between devices outside the vehicle), as well as the currently available datasets for each of these infrastructures. This study, therefore, will serve as a conceptual and referential guide for developing our solution.

Finally, focusing specifically on the security of autonomous vehicle systems, this study [Hidalgo et al. 2021] highlights how increasing the security and privacy of autonomous vehicles against dangerous cyberattacks will lead to a considerable reduction in the global number of deaths and injuries caused by accidents. Therefore, it is crucial to place significant emphasis on the security of communication for these vehicles and ensure their proper functioning, even while under attack, as they represent a high-risk system. In this context, the study introduces a project called SerIoT, which provides a strategy for real-time monitoring of traffic between different IoT platforms used by autonomous vehicle systems. According to the authors, this tool is capable of recognizing suspicious patterns, evaluating them, and ultimately providing decisions to mitigate such actions. Additionally, the authors employ a GNN framework to detect attacks within the network, with a metric evaluation of the results that could serve as a reference for analyzing the model in our work. Notably, the article also demonstrates several strategies that can be


**Figure 5. DARE dataset format. [HAIDAR et al. 2020]**

applied within the VANET context to prevent the consequences of attacks on the system, which will undoubtedly be highly valuable for our research as well.

## 4. Applied Methodology

This section describes the steps and decisions taken to implement the experimental part of this project. The concept of modeling the network in a graph form followed the previously explained host-connected proposal provided by [Pujol-Perich et al. 2021a] (Figure 4). This way of representing the graph including heterogeneous elements (i.e., hosts and flows) led the authors to devise a new message-passing architecture specifically adapted to process and learn the host-connection graph features (Figure 6). Each node has its own hidden state (features vector), which is refreshed (update step) considering the data accumulated in the gather function (aggregation step). The aggregation step simply employs a compression function to concatenate the hidden states of two connected nodes. Afterward, the hidden states are updated considering the information collected in the new aggregated message. This is done by applying the update function to the aggregated message and the current hidden state of the node. As a result, all of the functions used in both steps are learnable functions that can be approximated by neural networks during training. Particularly, the authors implemented the compression functions of the aggregation step as 2-layer fully connected NNs, while the update functions are modeled as Gated Recurrent Units (GRUs [Chung et al. 2014]). Finally, the readout function takes as input the final hidden states of each flow and outputs the predicted class for the flow (either a specific attack or benign traffic). This function is implemented with a 3-layer fully connected NN, where all the possible output classes are represented via one-hot encoding.

For the experimental part, we used a publicly available dataset named DARE [DARE-dataset 2024]. This dataset provides details about the specific VANET transmission system of interest to us. It includes a large number of misbehavior reports encoded in JavaScript Object Notation (JSON), as shown in Figure 5. Therefore, we processed the data to suit the required format as an input for our algorithm. First, we selected the most important features for our work: receiver and sender vehicle ID, attack label, and the BSM messages content (GPS coordinates, position, speed, acceleration, heading). Then, we filtered the data to consider only the DoS types of attacks, since it contained a group of eight different malicious anomalies. Lastly, we divided the filtered file into random

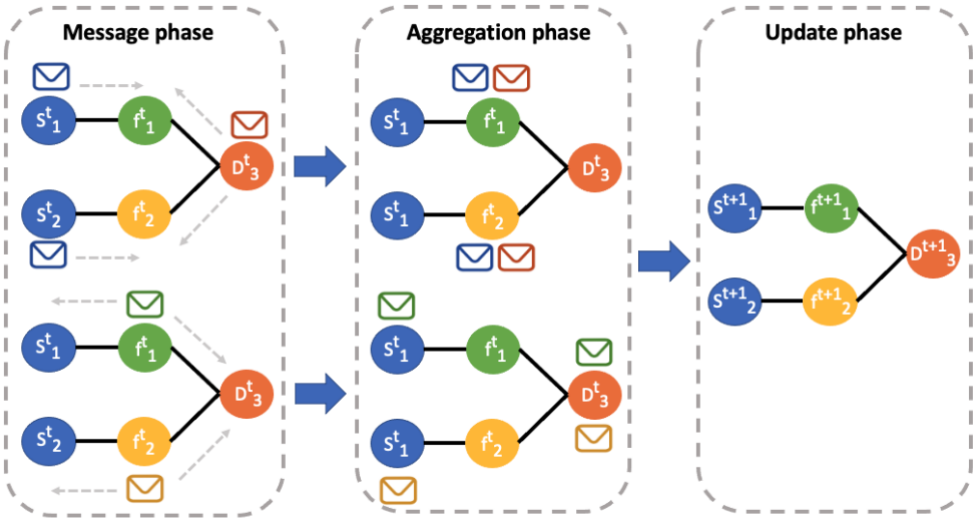


**Figure 6. Illustration of the message-passing phase for the host-connection proposed graph. [Pujol-Perich et al. 2021a]**

splits of 80% and 20% samples for the training and validation steps of the algorithm, respectively, and converted it to Comma-Separated Values (CSV) format. All the actions conducted in this part of the project were performed under the operation of the pandas library written for the Python language.

## 5. Evaluation

The evaluation procedure took into account the selection of various metric performances and different state-of-the-art ML-based NIDS algorithms for comparison purposes (Decision Tree, Random Forest, XGBoost and Multi-Layer Perceptron). These classifiers were deployed using the scikit-learn library and trained using the default parameters of the individual functions. To assess the performance of the algorithms over the classification task, we used the traditional measurements of a predictive model: accuracy, recall, precision, macro-F1, weighted-F1, and specificity. From that, we selected only the accuracy and the weighted-F1 as comparable scores between the models, since they unify in a single metric the precision and recall measures of the solutions. Moreover, we also chose the inference time and computational cost (CPU and RAM resources usage) as an evaluation of the efficiency metrics for each model. The inference time calculation was performed measuring the interval time for the models to predict all the sample inputs in the validation data. In conjunction with this task, the computational cost was quantified tracing the information retrieved by the *top* Linux command on terminal.

At last, we also selected some explainability metrics of a GNN architecture, in order to assist in interpreting the generic ability of the model to correctly classify unseen data. According to [Yuan et al. 2022], there are three recently proposed evaluation metrics for explanation tasks: fidelity, sparsity, and stability. Fidelity and sparsity identify input features that are important for the model and ignore the irrelevant ones. Fidelity is defined as the difference of accuracy (or predicted probability) between the original predictions and the new predictions after masking out important input features. Sparsity measures the fraction of features selected as important by explanation methods. For both

$$Fidelity+^{acc} = \frac{1}{N}\sum_{i=1}^{N}(\mathbb{1}(\hat{y}_i = y_i) - \mathbb{1}(\hat{y}_i^{1-m_i} = y_i)),$$

$$Sparsity = \frac{1}{N}\sum_{i=1}^{N}(1 - \frac{|m_i|}{|M_i|}),$$

**Figure 7. Formulas used for the explainable metrics [Yuan et al. 2022]. Here, $y_i$ is the original prediction of graph i and N is the number of graphs. 1 - $m_i$ means the complementary mask that removes the important input features. The indicator function 1($\hat{y}i$ = $y_i$) returns 1 if $\hat{y}i$ and $y_i$ are equal and returns 0 otherwise. $m_i$ denotes the number of important input features (nodes/edges/node features) identified in $m_i$ and $M_i$ means the total number of features.**

metrics, higher values indicate better explanation results. Figure 7 depicts the formulas used for each metric.

The stability metric measures whether an explanation method is stable. The key idea of stability is to apply small changes to the input graph without affecting the predictions and measure the difference between the results. Hence, we decided to artificially increase the value in some features of the training dataset and reevaluate the model with the validation dataset. Specifically, we randomly selected 15% of the training dataset, added an arbitrary constant value to its features, and trained the model again with the same hyperparameters and metrics. We also retrained the other classifier models and compared the results to evaluate the difference of robustness among them.

## 5.1. Results

All the training and development process was conducted on a computer having 13th Gen Intel(R) Core(TM) i5-1334U CPU with 4.60 GHz processing frequency, 16GB of main memory and the machine is operated with 64-bit Ubuntu 22.04 LTS operating system. The GNN algorithm was created using the tensorflow framework for machine learning and the metrics were measured using the open-source Keras library that provides a Python interface for artificial neural networks. For the first experiment, we trained the algorithm over 100 epochs, whereas each epoch contained 1000 steps (iterations over the data). The model hyperparameters used for the implementation design and performance evaluation process are shown in Table 1. Table 2 shows the results reported by the final step of some epochs. The rows of the table representing the metrics are presented as follows:

**Table 1. GNN hyperparameters used during the development process**

| Parameter Name | Value |
|---|---|
| Loss Function | Binary Cross Entropy |
| Optimizer | Adam |
| Learning rate | 0.001 |
| Number of epochs | 100 |
| Steps per epoch | 1000 |
| Validation steps | 100 |
| Batch size | 16 |

**Table 2. Results of the GNN algorithm trained with 100 epochs**

| Epochs | 1st epoch | 10th epoch | 20th epoch | 50th epoch | 100th epoch |
|---|---|---|---|---|---|
| Loss | 0.5146 | 0.1651 | 0.1412 | **0.0773** | 0.0878 |
| Categorical Accuracy | 0.7317 | 0.9491 | 0.9597 | 0.9772 | **0.9779** |
| Specificity at sensitivity | 0.9956 | 0.9966 | 0.9968 | **0.9979** | 0.9974 |
| Recall 0 | 0.4918 | 0.9326 | 0.9455 | **0.9633** | 0.9617 |
| Precision 0 | 0.5628 | 0.9025 | 0.9247 | 0.9619 | **0.9665** |
| Recall 1 | 0.8352 | 0.9562 | 0.9660 | 0.9833 | **0.9852** |
| Precision 1 | 0.7921 | 0.9703 | 0.9757 | **0.9839** | 0.9830 |
| Macro F1 | 0.6690 | 0.9403 | 0.9529 | 0.9731 | **0.9741** |
| Weighted F1 | 0.7262 | 0.9493 | 0.9599 | 0.9772 | **0.9779** |

- **Loss:** expected error generated by the utilized loss function (binary cross-entropy).
- **Categorical Accuracy:** measures the proportion of correctly predicted classes among all predicted classes. In other words, it calculates how often the predicted output matches the true output.
- **Specificity at sensitivity:** computes best specificity where sensitivity is greater or equal than 0.1 (specified value chosen for our model). Sensitivity, also known as recall, measures how well a model can detect positive instances. Specificity measures how well a model can detect negative instances.
- **Recall 0:** recall metric for the DoS label (proportion of DoS attacks that are correctly identified by the model).
- **Precision 0:** precision metric for the DoS label (proportion of DoS attacks that are correctly predicted by the model).
- **Recall 1:** recall metric for the benign label (proportion of benign flows that are correctly identified by the model).
- **Precision 1:** precision metric for the benign label (proportion of benign flows that are correctly predicted by the model).
- **Macro F1:** F1 score calculated by unweighted average of precision and recall.
- **Weighted F1:** F1 score calculated by weighted average of precision and recall. The weight used for each label is the number of true instances.

As we can see from Table 2, the metrics used to evaluate our model produced great performance, since every measure at the end of the 100th epoch (last column of the table) was greater than 96%. It is important to realize how the metrics evolved over time, generating better results for each step of the epochs. The biggest modifications came on between epochs first and tenth, where, for example, the accuracy changed from 0.7317 to 0.9491 and the Recall 0 metric jumped from 0.4918 to 0.9326. The best results position in between the 50th and 100th epochs, as highlighted in the table. However, the results difference among these two parts usually varies at the last decimal places, which represents a small impact in the solution. Therefore, we selected the 100th epoch as the stopping step in the training phase.

For the second experiment, we used the same dataset samples to train other ML classifiers and compare the results. Figure 8 shows the results of this experiment, which measured initially only the predictive metrics. The GNN model achieved the best results among the classifiers, alongside the Random Forest algorithm. Then, we also measured

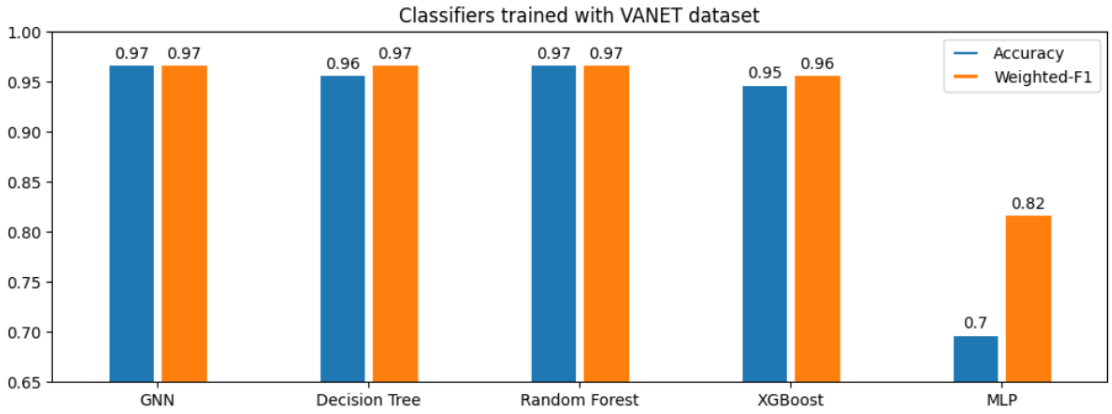


**Figure 8. Comparison of different ML models trained with the same data**

**Table 3. Performance metrics of each model measured using the validation dataset**

| Classifier | Inference time (s) | CPU usage (%) | RAM usage(%) |
|---|---|---|---|
| GNN | 5.69 | 415.3 | 5.4 |
| **Decision Tree** | **1.43** | **50.2** | **1.6** |
| Random Forest | 53.79 | 106.7 | 1.8 |
| XGBoost | 6.63 | 1195 | 2.0 |
| MLP | 1.65 | 1196 | 2.1 |

the computational cost and inference time of each model to evaluate its performance over the classification task. Table 3 displays the results for this experiment. In some cases, the CPU usage surpassed the 100% threshold because the machine CPU has 12 cores, and, therefore, the actual limit is 1200%. The Decision Tree model appears to be the most lightweight one, as it is also the most simple. Since the GNN model is trained over 100 epochs, it is expected to reach high levels of computational cost. Additionally, the inference time presented a reasonable result compared to the other cases.

In the third experiment, we measured the explainability metrics for a graph model. Figure 9 illustrates the results achieved in this part. As mentioned above, higher values for both metrics indicate a good sign, proving that the graph is evolving over the training phase to consider only the features (nodes and edges) that are more relevant to the predictive task. A higher value for the fidelity metric suggests that discriminative features are being identified by the model. On the other hand, higher values for the sparsity metric show that the explanations are more sparse and tend to only capture the most important input information. Thereby, these patterns can be observed in our model according to Figure 9, as both metrics are increasing over time.

In the end, we performed our last experiment of artificially modifying the training dataset and observing the impacts of this change in the metrics, in order to evaluate the robustness and stability of the models. Table 4 demonstrates the results collected in this experiment for the GNN model. The results show little difference between Table 2, when the algorithm was trained with the regular data, and seemingly tend to be converging over the last epochs. The most significant alterations were noted amid the Recall 0 and Precision 0 metrics, that represent the predictive values for the DoS attack label, where they decreased from 0.9617 to 0.9456 and from 0.9665 to 0.9373, respectively.

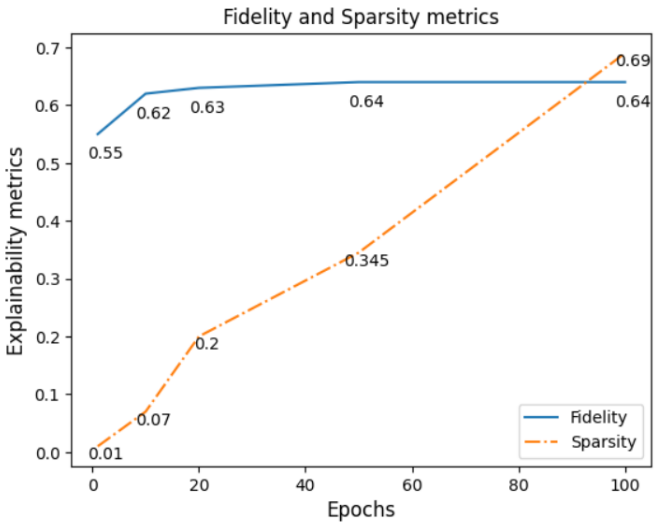


**Figure 9. Explainable GNN metrics measured during the training phase.**

Nonetheless, they still imply good performances for the prediction task. This represents that the noises artificially generated in the data did not impact the predictive ability of the model, meaning it holds great stability and is robust to perturbations in unpredictable situations. Besides that, we also utilized the same noisy data to evaluate the performance of the other models in this scenario. Figure 10 describes the results of this process. Not surprisingly, all the other classifiers decreased their corresponding metrics, except for the MLP model, which already had lower measurements in the past. Meanwhile, the GNN model maintained its high values, decreasing merely 0.01 points for both metrics.

**Table 4. Results of the GNN algorithm trained with artifical noisy data**

| Epochs | 1st epoch | 10th epoch | 20th epoch | 50th epoch | 100th epoch |
|---|---|---|---|---|---|
| Loss | 0.3971 | 0.1962 | 0.1625 | 0.1353 | **0.1141** |
| Categorical Accuracy | 0.8385 | 0.9306 | 0.9478 | 0.9604 | **0.9643** |
| Specificity at sensitivity | 0.9949 | 0.9971 | 0.9966 | 0.9973 | **0.9979** |
| Recall 0 | 0.7340 | 0.9061 | 0.9151 | 0.9340 | **0.9456** |
| Precision 0 | 0.7333 | 0.8703 | 0.9134 | 0.9354 | **0.9373** |
| Recall 1 | 0.8839 | 0.9413 | 0.9621 | 0.9718 | **0.9725** |
| Precision 1 | 0.8843 | 0.9585 | 0.9629 | 0.9712 | **0.9763** |
| Macro F1 | 0.8089 | 0.9188 | 0.9384 | 0.9531 | **0.9579** |
| Weighted F1 | 0.8385 | 0.9310 | 0.9478 | 0.9603 | **0.9644** |

Finally, we discuss some limitations and potential improvements of our project. The biggest constraint of this work lies on the dataset weaknesses. In addition to being a relatively old dataset produced in 2020, in order to increase the input amount of data (number of observations) for our algorithm, we joined some of the DoS attacks sub-classes, such as Distributed DoS (DDoS), DoS-Disruptive, DoS-DataReplay, etc., and labeled them as pure DoS. This could eventually lead to a wrong evaluation of the model in some cases, since the patterns differ from each other in some of these attacks. Thus, we should consider using different attacks' classes in future works and note whether the GNN model keeps its same effectiveness. Furthermore, it is worthwhile mentioning that there

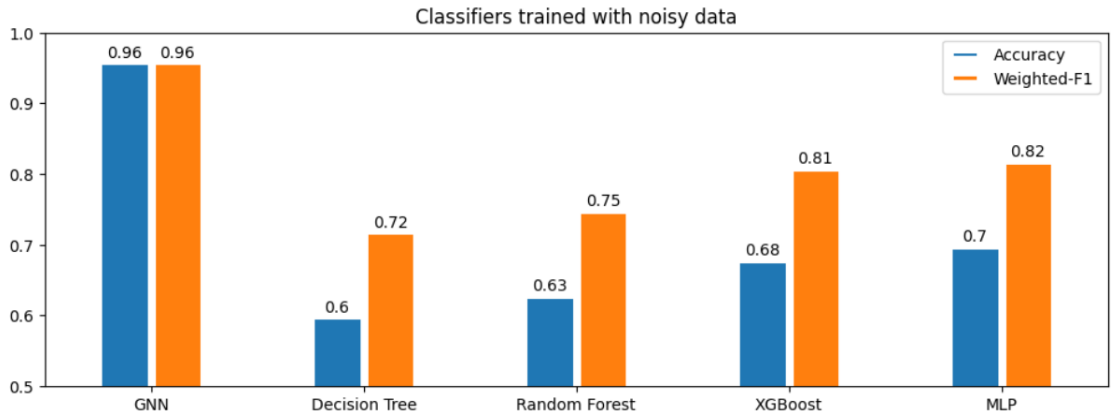


**Figure 10. Different ML models trained with artificial noisy data**

is a strong demand for concrete and public datasets regarding the VANET environment, as we had difficulty finding the desired dataset for our specific VANET system. This way, the use of simulation tools to enhance the quality of the data is well accepted for future developments of this work.

## 6. Conclusion

Solutions for intrusion detection in autonomous vehicles are essential to prevent a failure of these systems. While a lot of Machine Learning techniques were invented over the past few decades and tested for this specific context, most of them failed to come up with a good generalization method for the application scenarios in real life. To overcome this constraint, recently researchers have been examining the power of GNN models to generalize network problems involving cyber attacks. This work intended to join these two areas, which is something scarce, if not unique, among the academic work focused on the VANETs scope. The GNN evaluation carried out by analytic modeling took into account the public DARE dataset and the results achieved confirmed the remarks pointed out by the recent studies. Our model exceeded all the other traditional ML techniques for intrusion detection and presented a great robustness towards eventual modifications in the network features. For future works, we intend to improve the study using simulation tools, such as NS3, SUMO, and Omnet++, but also experiment with novel GNN algorithms by applying newly reviewed concepts in other works.

## References

Ahmed, W. and Elhadef, M. (2019). Dos attacks and countermeasures in vanets. *Advanced Multimedia and Ubiquitous Engineering - Springer*, 518:333–341.

Alam, M., Ferreira, J., and Fonseca, J. (2016). Introduction to intelligent transportation systems. *Studies in Systems, Decision and Control - Springer*, pages 19250–19276.

Baccari, S., Hadded, M., Ghazzai, H., Touati, H., and Elhadef, M. (2024). Anomaly detection in connected and autonomous vehicles: A survey, analysis, and research challenges. *IEEE Access*, 12:19250–19276.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Deep Learning and Representation Learning Workshop*.

Corona, I., Giacinto, G., and Roli, F. (2013). Adversarial attacks against intrusion detection systems: Taxonomy, solutions and open issues. *Information Sciences - Elsevier*, 239:201–225.

da Silva, E. S., Pedrini, H., and Santos, A. (2023). Applying graph neural networks to support decision making on collective intelligent transportation systems. *IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT*.

DARE-dataset (2024). `https://github.com/josephkamel/DARE-Dataset`. [Online; accessed Oct 2024].

Franceschi, L., Niepert, M., Pontil, M., and He, X. (2019). Learning discrete structures for graph neural networks. *36th International Conference on Machine Learning (ICML)*.

Hadded, M., Muhlethaler, P., Laouiti, A., Zagrouba, R., and Saidane, L. A. (2015). Tdma-based mac protocols for vehicular ad hoc networks: A survey, qualitative analysis, and open research issues. *IEEE Communications Surveys and Tutorials*, 17(4):2461–2492.

HAIDAR, F., KAMEL, J., Jemaa, I. B., Kaiser, A., LONC, B., and Urien, P. (2020). Dare: A reports dataset for global misbehavior authority evaluation in c-its. *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*.

Hidalgo, C., Vaca, M., Nowak, M. P., Frölich, P., Reed, M., Al-Naday, M., Mpatziakas, A., Protogerou, A., Drosou, A., and Tzovaras, D. (2021). Detection, control and mitigation system for secure vehicular communication. *Elsevier Inc.*

Kriege1, N. M., Johansson, F. D., and Morris1, C. (2020). A survey on graph kernels. *Applied Network Science - Published by Springer*, 5.

Lamnabhi-Lagarrigue, F., Annaswamy, A., Engell, S., Isaksson, A., Khargonekar, P., Murray, R. M., Nijmeijer, H., Samad, T., Tilbury, D., and den Hof, P. V. (2017). Systems and control for the future of humanity, research agenda: Current and future roles, impact and grand challenges. *Annual Reviews in Control*, 43:1 – 64.

Limbasiya, T., Teng, K. Z., Chattopadhyay, S., and Zhou, J. (2022). A systematic survey of attack detection and prevention in connected and autonomous vehicles. *Vehicular Communications - Published by Elsevier*, 37.

Lu, N., Cheng, N., Zhang, N., Shen, X., and Mark, J. W. (2014). Connected vehicles: Solutions and challenges. *IEEE Internet of Things Journal*, 1:289–299.

Masmoudi, M., Ghazzai, H., Frikha, M., and Massoud, Y. (2019). Object detection learning techniques for autonomous vehicle applications. *2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*.

McClelland, J. L. and Rumelhart, D. E. (1986). *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*. The MIT Press.

MITRA, S., CHAKRABORTY, T., NEUPANE, S., PIPLAI, A., and MITTAL, S. (2024). Use of graph neural networks in aiding defensive cyber operations. *ACM Trans. Priv. Sec.*

Nagarajan, J., Mansourian, P., Shahid, M. A., Arunita, J., Saini, I., Zhang, N., and Kneppers, M. (2023). Machine learning based intrusion detection systems for connected

autonomous vehicles: A survey. *Peer-to-Peer Networking and Applications*, 16:2153–2185.

Pujol-Perich, D., Suárez-Varela, J., Cabellos-Aparicio, A., and Barlet-Ros, P. (2021a). Unveiling the potential of graph neural networks for robust intrusion detection. *Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain*.

Pujol-Perich, D., Suárez-Varela, J., Cabellos-Aparicio, A., and Barlet-Ros, P. (2021b). Unveiling the potential of graph neural networks for robust intrusion detection. *Barcelona Neural Networking Center, Universitat Politècnica de Catalunya, Spain*.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2008). The graph neural network model. *IEEE Transactions on Neural Networks*, 20:61 – 80.

Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *2010 IEEE Symposium on Security and Privacy*.

Wang, J., Zhang, L., Huang, Y., and Zhao, J. (2020). Safety of autonomous vehicles. *Journal of Advanced Transportation*.

Wu, L., Lin, H., Gao, Z., Tan, C., and Stan.Z.Li (2023). Self-supervised learning on graphs: Contrastive, generative, or predictive. *IEEE Transactions on Knowledge and Data Engineering*, 35:4216 – 4235.

Xian, X., Wu, T., Ma, X., Qiao, S., Shao, Y., Wang, C., Yuan, L., and Wu, Y. (2022). Generative graph neural networks for link prediction. *arXiv preprint arXiv:2301.00169*.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? *2019 International Conference on Learning Representations (ICLR)*.

Yuan, H., Yu, H., Gui, S., and Ji, S. (2022). Explainability in graph neural networks: A taxonomic survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.