Applying Graph Neural Networks to Identify Denial of Services Attacks in Vehicular Ad-Hoc Networks

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)
June 2024

{samuelhma91,aldri}@dcc.ufmg.br

Abstract. *Cyber-security 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 and prevention of these attacks. Nevertheless, they still can not keep up with the fast-paced nature of these adversarial attacks. In this work, we are proposing to study a trendy method called Graph Neural Networks to evaluate its efficiency and robustness over these challenges, specifically in the self-driving vehicles environment, using a publicly available dataset. The preliminary results showed that the measured metrics achieved a great performance, which paves the way for future works looking to assess stronger variables and sophisticated scenarios.*

1. Introduction

Given the upward usage of the Internet in the various services of society, people and devices are increasingly becoming more reliant on online access [Chase 2013]. However, this connectivity, despite all of its provided benefits, leaves the users exposed to an environment full of malicious actors seeking to take advantage of the security failures for their own profits. Besides that, the systems are increasingly becoming more complex and dynamic, hindering the handling of the data and a reliable communication over their components [Ammar et al. 2018]. That way, in order to warrant the correct use of the applications across the internet, it is required to maintain the full security of the network using modern solutions.

In the face of 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 monitor 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, although all of the tests and investments that are being conducted in this area, there are still a significant number of obstacles that impede the delivery of this technology to society , demanding a deeper evaluation of these problems.

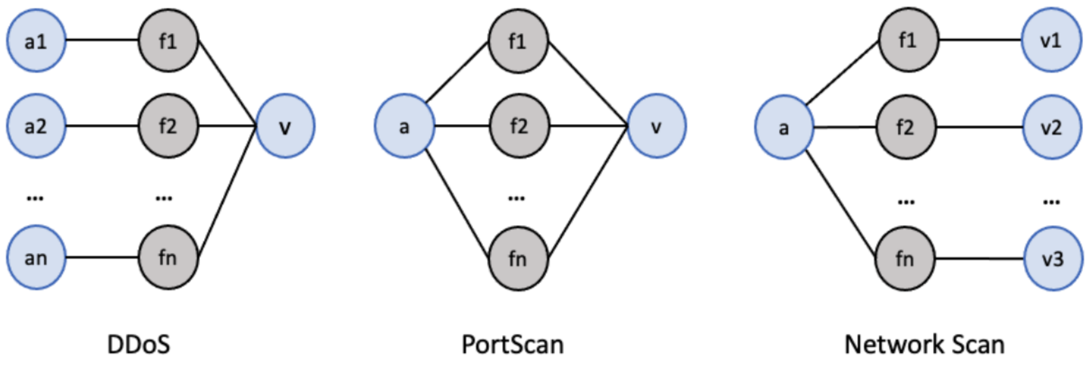


Figure 1. Cyber-attacks flows represented as graphs. Image taken from [Pujol-Perich et al. 2021]

Following these lines, a lot of researches over the years have investigated the use of Machine Learning methods to assist in the protection of systems. Even though they achieved a great accuracy, most of the times they were not considered for use in the real-world application' software [Sommer and Paxson 2010]. This can be explained because the studied models underlie its analysis from the network traffic data to find features that allow the recognition of the attacks [Khraisat et al. 2019] . Since these models were trained over a dataset of specific flow-level features, we can imagine this generated a high degree of over-fitting and, consequently, a lack of generalization over traffic of other networks. Considering the context of computer networks, the applications in the 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, despite the good results during the experiments, these models failed to find an appropriate generalization to the real world, which makes them strongly vulnerable to traffic changes and adversarial attacks [Pujol-Perich et al. 2021].

Therefore, 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. One of the ways to achieve it is to model the network as a graph, as this allows visualization of the arrangement of the elements and, thus, to apply algorithms on the graph topology as a way of examining the structure of the network and the relationship of its components [Wu et al. 2024]. Unlike the features from the traffic data, the attack's patterns cannot be modified by malicious agent, which is a better way to tackle this issue.

Recently, a lot of studies have been focusing on using Graph Neural Networks (GNN) to assist in this task. The idea consists of shaping the network elements in a graph with edges and nodes [Busch et al. 2021] and classify its feature patterns into a benign or malignant flow. We will follow that approach in this work, considering the context of Vehicular Ad-Hoc Networks (VANETs). The essential part of this idea handles to capture not only the individual features of flows, but also their relationships within the network. This way, we expect the model to learn the underlying structural flow patterns of attacks and achieve a deeper knowledge and characterization of them.

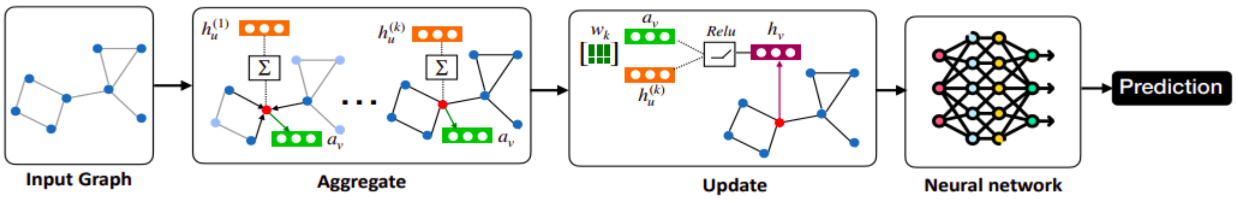


Figure 2. Training steps of a GNN. Image taken from [MITRA et al. 2024]

2. Background

In our project, we are interested in applying GNN methods to support the cyber-security 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 into the Artificial Intelligent 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 rely 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 to capture 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 GNN framework: (1) aggregation and (2) update (Figure 2). 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 UP-

DATE(·) functions created, such as Graph Convolutional Networks (GCN) and Graph Attention Network (GAT).

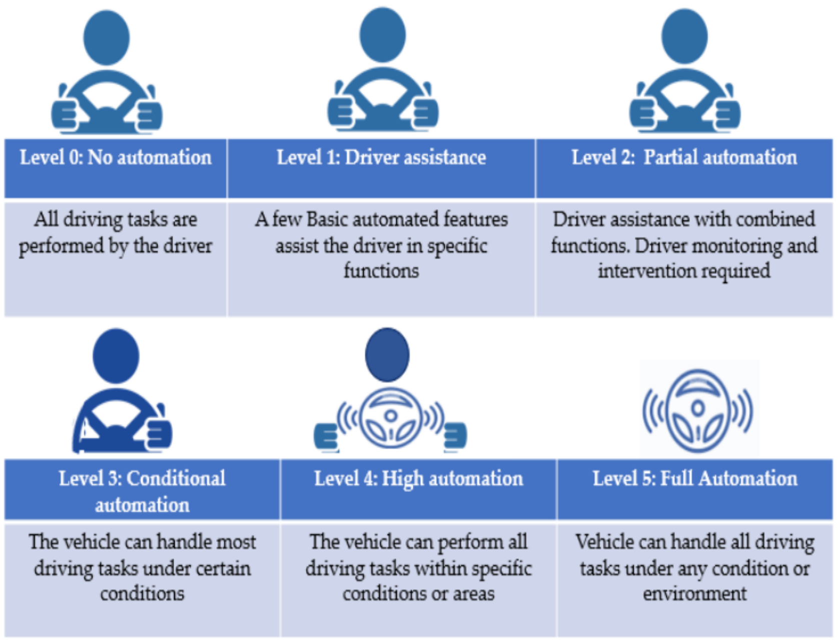
One of the popular approaches to learning with graph-structured data is to make use of graph kernels (functions that measure the similarity between graphs) plugged into a kernel machine, such as a support vector machine [Kriege1 et al. 2020]. Kernel methods refer to machine learning algorithms that learn 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 GNN, 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 (Figure 3) [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 his 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.

One of the ways these systems ensure road safety is through the adoption of a Cooperative Awareness Service (CAS), which transmits crucial information to designated receivers regarding the position, motion state, and dimension of an Intelligent Transportation System Station (ITS-S) [ITS 2019]. Meanwhile, the Collective Perception Service (CPS) has been developed to facilitate the exchange of information between Intelligent Transportation Systems (ITS) devices and nearby receivers, enabling them to share their surroundings. By improving situational awareness, CPS has the potential to greatly enhance road safety, reduce traffic congestion, and optimize traffic management [Baccari et al. 2020]. Artificial intelligence and machine learning techniques have frequently been suggested as potential solutions for improving the efficiency of Collective



Level 0: No automation	Level 1: Driver assistance	Level 2: Partial automation
All driving tasks are performed by the driver	A few Basic automated features assist the driver in specific functions	Driver assistance with combined functions. Driver monitoring and intervention required

Level 3: Conditional automation	Level 4: High automation	Level 5: Full Automation
The vehicle can handle most driving tasks under certain conditions	The vehicle can perform all driving tasks within specific conditions or areas	Vehicle can handle all driving tasks under any condition or environment

Figure 3. Driving automation levels. Image taken from [Baccari et al. 2024]

Intelligent Transportation Systems (CITSs), with a particular emphasis on the Collective Perception Service (CPS) [Tang et al. 2015].

An automated vehicle (AV) consists of several key components that can be organized into three layers: sensing, perception, and decision layers [Deng et al. 2021]. These components operate collaboratively to allow the vehicle to sense and understand its environment, make appropriate decisions, and navigate smoothly on the roads. Thus, as a first step, the sensors take care of collecting data from the environment. These data are then processed in the second layer in order to extract relevant information, such as recognizing objects, identifying obstacles, and determining their positions. The extracted information is subsequently used to generate commands. Finally, the decision layer takes responsibility for translating orders into mechanical actions such as braking, acceleration, and steering [Malik et al. 2022] (Figure 4).

Due to sensor data uncertainties caused by either cyber attacks or other external factors, such as sensor malfunctions, environmental anomalies and weather conditions, autonomous driving systems require a sophisticated design to capture those abnormalities and eventually mitigate their impacts [Wang et al. 2020]. Nevertheless, despite all the efforts that have been made in this sector, there are still a number of significant obstacles that prevent the widespread use of this technology. One of the primary barriers to its wider adoption is anomalies and unanticipated happenings [Liu et al. 2021]. Because of this, anomaly detection is an essential task to guarantee the safety of autonomous vehicles and the certainty of their decisions. In general, an anomaly, commonly referred to as an outlier or corner case, occurs when a measurement or reading significantly diverges from the typical values generated by a sensor. In simpler terms, it represents data that deviates from the rule and shows unexpected behavior compared to what the sensor usually produces [Cook et al. 2020].

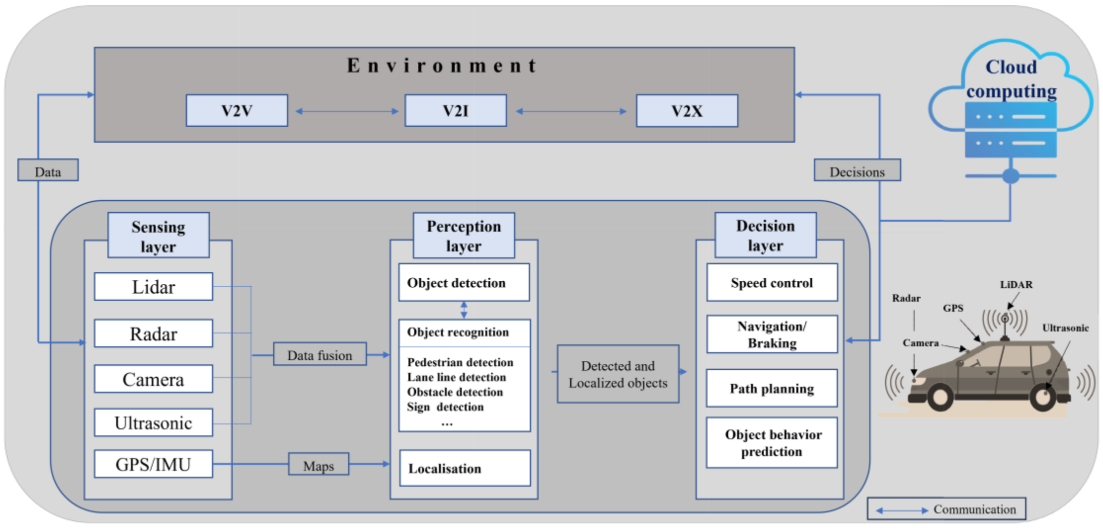


Figure 4. Architecture of an automated driving system. Image taken from [Baccari et al. 2024]

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. Thus, several 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 that are going to be used in the experiment 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 work [MITRA et al. 2024], which will also serve as a guide for this research, the authors studied in detail how the application of GNNs can help break down the various phases within the stages of the lifecycle of an attack, more specifically a famous attack called CKC (Lockheed Martin Cyber Kill Chain). Therefore, the study can serve as a basis to guide this research by providing a model for the application of the same approaches within cyber-attacks on vehicular networks. The authors highlight how, among the possible Machine Learning techniques, the GNN model has grown as a promising alternative for strengthening the effectiveness of defense measures necessary for attack prevention in a system. This is mainly due to the model's ability to process and learn from various types of data present in cyber threats. However, the authors conclude that, despite all the benefits that the use of this model provides, it is important to remember that the cyberspace is dynamic and constantly changing, which makes it necessary for even GNNs to adapt to these changes. This highlights the need to continue with further studies that enable the exploration of new horizons in this area.

Similarly, within the approach of the autonomous vehicle environment, the study by [Baccari et al. 2024] analyze the current scenario of research in this field, with special attention to solutions for detecting anomalies in sensor data. They emphasize that one of the main challenges for the proper functioning of vehicles lies in the reliability of data coming from sensors, as they provide important information for system decision-making, which can consequently result in catastrophic failures. Such sensors are vulnerable to different types of anomalies, resulting, for example, from adverse weather conditions, technical problems, and cyber-attacks. Therefore, as a way to contribute to future studies, the researchers identify in their conclusion of the results what are the main points that need to be further explored in future research, in order to develop anomaly detection systems that increase the reliability of autonomous vehicles.

Finally, specifically addressing the security of autonomous vehicle systems, the authors in [Hidalgo et al. 2021] show how increasing the security and privacy of autonomous vehicles against dangerous cyber-attacks will lead to a considerable reduction in the overall number of deaths and injuries caused by accidents. Therefore, it is important to have a very strong focus on the security of vehicle communication and ensure their proper functioning, even when they are under attack, as it is a high-risk system. Thus, the work presents a project, called SerIoT, which provides a strategy for monitoring real-time traffic between different IoT platforms. According to the authors, this system is capable of recognizing suspicious patterns, evaluating them, and finally providing decisions to mitigate these actions. Therefore, this project can serve as a reference for estimating the tests that will be performed on the GNN model and designing the desired performance for the various attack correspondences.

4. Methodology

This section describes the steps and decisions taken to implement the experimental part of this project. Essentially, we are interested in testing a simple GNN algorithm with a set of data generated by a vehicular autonomous network. The concept of modeling the problem (cyber attacks in VANETs) in a graph form will be followed using the host-connected proposal provided by [Pujol-Perich et al. 2021] (Figure 1). The available code of the project on GitHub was also utilized to test our approach, with some adaptations explained afterward. The authors suggested the representation of each flow as a node of a graph. Given a set of flows, a host-connection graph includes a node for each distinct host involved – either sending or receiving traffic. Thus, considering a flow f, with a source host S, and a destination host D, we create two undirected edges: 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 more straightforward representation would be to consider only hosts as node graphs, and flows as graph edges connecting the src/dst hosts. However, the decision to add specific nodes representing each flow was driven by the way GNN models operate. GNNs consider only as learnable objects the hidden states of nodes in input graphs. As a result, to properly learn embeddings on flows, it is needed to add them as nodes of the graph.

This way of representing the graph including heterogeneous elements (i.e., hosts and flows), which is not well supported by standard GNN models, led the authors to devise as well a new message-passing architecture specifically adapted to process and learn the host-connection graph features (Figure 5 below). 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 uses a compression function to concatenate the hidden states of two connected nodes, i.e., an edge in the input graph of the GNN. 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 aggregations step as 2-layer fully-connected NNs, while the updated functions are modeled as Gated Recurrent Units (GRUs [Chung et al. 2014]). Finally, it is necessary to define the readout function. It 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.

The data to train the model can be obtained in two ways: by making use of publicly available datasets or by generating our own datasets by simulating the VANET environment [Nagarajan et al. 2023]. Each approach has its own benefits and drawbacks. The simulation method is popular in the works done in VANET because of its flexibility to consider various scenarios. Despite the realistic and reliable results of the simulation method, using a dataset is more convenient because of its simplicity and lower computational power requirement. On the other hand, ready-finished datasets may not cover all of the scenarios required for our cases. Therefore, considering the initial scope of this project, it would be more suitable to test a public dataset, whereas the simulation can be used to create more precise scenarios for future works. Taking into account the needs of our context (identification of hosts, i.e., senders and receivers nodes), the only available dataset was found in this specific work [Gonçalves et al. 2020b]. The authors presented a dataset for VANET IDS, including both normal and attack data. This dataset was generated by NS3 as the network simulator and SUMO as the traffic simulator. The attacks implemented in this dataset are Denial of Service (DoS) and Fabrication attacks for speed, acceleration, and fake heading. Their proposed dataset is available for researchers at [Gonçalves et al. 2020a]. For our end, we will be using only the DoS data provided.

Since our dataset and the one used in the algorithm we are going to be based on are different, it is necessary to make some adaptations in the original code. Two main differences conducted the changes needed for the algorithm to work properly on the new

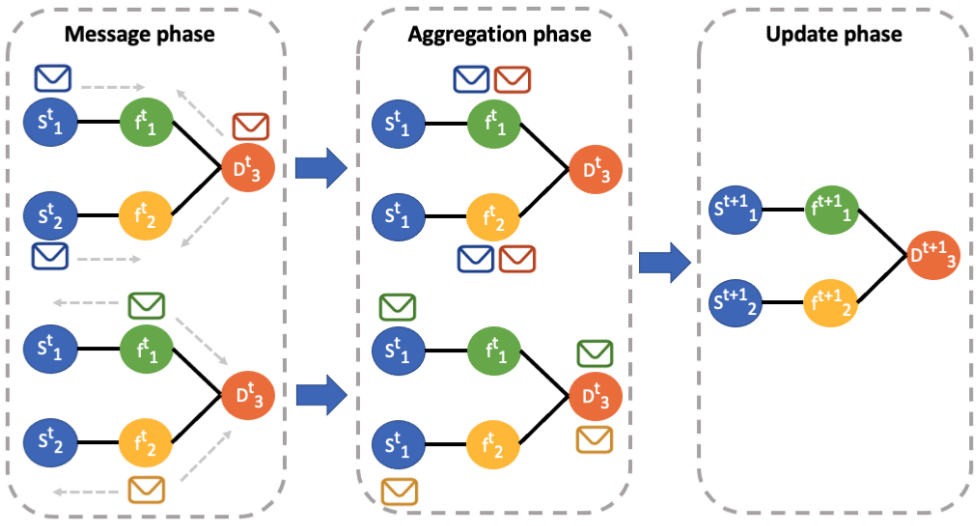


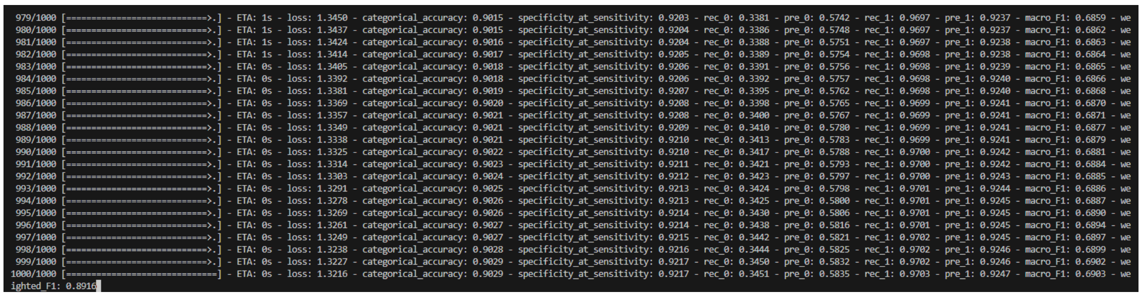
Figure 5. Illustration of the message-passing phase for the host-connection proposed graph. Image taken from [Pujol-Perich et al. 2021]

dataset: identification of the hosts and classification of the results. In the original code, the IPs of the hosts were considered as identifiers to construct the graph model. However, as we do not have IPs for the vehicles, we will simply use the vehicle's name identifiers located on the first two columns of the dataset (vehicle sender and vehicle receiver) for this purpose. The other required modification consisted of changing the final output of the neural network, since the original project proposed to classify 15 different attacks, but we are using only 2 labels (benign or DoS). Thus, we switched the *softmax* activation function of the last layer from a categorical cross-entropy loss function to a binary cross-entropy loss function instead. Lastly, as the environment context of the datasets was also different, we changed the data normalization statistics file to fit with the features of our dataset.

5. Evaluation, Results and Discussions

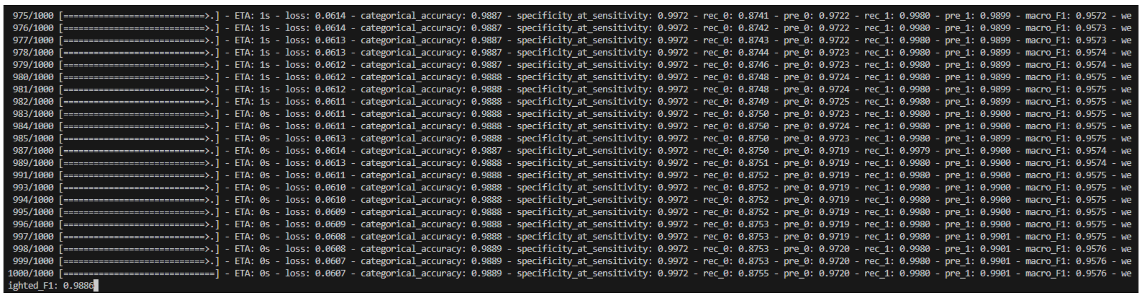
This section describes the results achieved by running the algorithm presented in the previous section with the preferred dataset. The model was trained over 10 epochs, whereas each epoch contained 1000 steps (iterations over the data). Figures 6 and 7 show examples of the outputs generated by the running code. Table 1 shows the results reported by the final step of each epoch. The columns of the table are presented as follows:

- **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).



```
 979/1000 [============================>.] - ETA: 1s - loss: 1.3450 - categorical_accuracy: 0.9015 - specificity_at_sensitivity: 0.9203 - rec_0: 0.3381 - pre_0: 0.5742 - rec_1: 0.9697 - pre_1: 0.9237 - macro_F1: 0.6859 - we
 980/1000 [============================>.] - ETA: 1s - loss: 1.3437 - categorical_accuracy: 0.9015 - specificity_at_sensitivity: 0.9204 - rec_0: 0.3386 - pre_0: 0.5748 - rec_1: 0.9697 - pre_1: 0.9237 - macro_F1: 0.6862 - we
 981/1000 [============================>.] - ETA: 1s - loss: 1.3424 - categorical_accuracy: 0.9016 - specificity_at_sensitivity: 0.9204 - rec_0: 0.3388 - pre_0: 0.5751 - rec_1: 0.9697 - pre_1: 0.9238 - macro_F1: 0.6863 - we
 982/1000 [============================>.] - ETA: 1s - loss: 1.3414 - categorical_accuracy: 0.9017 - specificity_at_sensitivity: 0.9205 - rec_0: 0.3389 - pre_0: 0.5754 - rec_1: 0.9698 - pre_1: 0.9238 - macro_F1: 0.6864 - we
 983/1000 [============================>.] - ETA: 0s - loss: 1.3405 - categorical_accuracy: 0.9018 - specificity_at_sensitivity: 0.9206 - rec_0: 0.3391 - pre_0: 0.5756 - rec_1: 0.9698 - pre_1: 0.9239 - macro_F1: 0.6865 - we
 984/1000 [============================>.] - ETA: 0s - loss: 1.3392 - categorical_accuracy: 0.9018 - specificity_at_sensitivity: 0.9206 - rec_0: 0.3392 - pre_0: 0.5757 - rec_1: 0.9698 - pre_1: 0.9240 - macro_F1: 0.6866 - we
 985/1000 [============================>.] - ETA: 0s - loss: 1.3381 - categorical_accuracy: 0.9019 - specificity_at_sensitivity: 0.9207 - rec_0: 0.3395 - pre_0: 0.5762 - rec_1: 0.9698 - pre_1: 0.9240 - macro_F1: 0.6868 - we
 986/1000 [============================>.] - ETA: 0s - loss: 1.3369 - categorical_accuracy: 0.9020 - specificity_at_sensitivity: 0.9208 - rec_0: 0.3398 - pre_0: 0.5765 - rec_1: 0.9699 - pre_1: 0.9241 - macro_F1: 0.6870 - we
 987/1000 [============================>.] - ETA: 0s - loss: 1.3357 - categorical_accuracy: 0.9021 - specificity_at_sensitivity: 0.9208 - rec_0: 0.3400 - pre_0: 0.5767 - rec_1: 0.9699 - pre_1: 0.9241 - macro_F1: 0.6871 - we
 988/1000 [============================>.] - ETA: 0s - loss: 1.3349 - categorical_accuracy: 0.9021 - specificity_at_sensitivity: 0.9209 - rec_0: 0.3410 - pre_0: 0.5780 - rec_1: 0.9699 - pre_1: 0.9241 - macro_F1: 0.6877 - we
 989/1000 [============================>.] - ETA: 0s - loss: 1.3338 - categorical_accuracy: 0.9021 - specificity_at_sensitivity: 0.9210 - rec_0: 0.3413 - pre_0: 0.5783 - rec_1: 0.9699 - pre_1: 0.9241 - macro_F1: 0.6879 - we
 990/1000 [============================>.] - ETA: 0s - loss: 1.3325 - categorical_accuracy: 0.9022 - specificity_at_sensitivity: 0.9210 - rec_0: 0.3417 - pre_0: 0.5788 - rec_1: 0.9700 - pre_1: 0.9242 - macro_F1: 0.6881 - we
 991/1000 [============================>.] - ETA: 0s - loss: 1.3314 - categorical_accuracy: 0.9023 - specificity_at_sensitivity: 0.9211 - rec_0: 0.3421 - pre_0: 0.5793 - rec_1: 0.9700 - pre_1: 0.9242 - macro_F1: 0.6884 - we
 992/1000 [============================>.] - ETA: 0s - loss: 1.3303 - categorical_accuracy: 0.9024 - specificity_at_sensitivity: 0.9212 - rec_0: 0.3423 - pre_0: 0.5797 - rec_1: 0.9700 - pre_1: 0.9243 - macro_F1: 0.6885 - we
 993/1000 [============================>.] - ETA: 0s - loss: 1.3291 - categorical_accuracy: 0.9025 - specificity_at_sensitivity: 0.9213 - rec_0: 0.3424 - pre_0: 0.5798 - rec_1: 0.9701 - pre_1: 0.9244 - macro_F1: 0.6886 - we
 994/1000 [============================>.] - ETA: 0s - loss: 1.3278 - categorical_accuracy: 0.9026 - specificity_at_sensitivity: 0.9213 - rec_0: 0.3425 - pre_0: 0.5800 - rec_1: 0.9701 - pre_1: 0.9245 - macro_F1: 0.6887 - we
 995/1000 [============================>.] - ETA: 0s - loss: 1.3269 - categorical_accuracy: 0.9026 - specificity_at_sensitivity: 0.9214 - rec_0: 0.3430 - pre_0: 0.5806 - rec_1: 0.9701 - pre_1: 0.9245 - macro_F1: 0.6890 - we
 996/1000 [============================>.] - ETA: 0s - loss: 1.3261 - categorical_accuracy: 0.9027 - specificity_at_sensitivity: 0.9214 - rec_0: 0.3438 - pre_0: 0.5816 - rec_1: 0.9701 - pre_1: 0.9245 - macro_F1: 0.6894 - we
 997/1000 [============================>.] - ETA: 0s - loss: 1.3249 - categorical_accuracy: 0.9027 - specificity_at_sensitivity: 0.9215 - rec_0: 0.3442 - pre_0: 0.5821 - rec_1: 0.9702 - pre_1: 0.9245 - macro_F1: 0.6897 - we
 998/1000 [============================>.] - ETA: 0s - loss: 1.3238 - categorical_accuracy: 0.9028 - specificity_at_sensitivity: 0.9216 - rec_0: 0.3444 - pre_0: 0.5825 - rec_1: 0.9702 - pre_1: 0.9246 - macro_F1: 0.6899 - we
 999/1000 [============================>.] - ETA: 0s - loss: 1.3227 - categorical_accuracy: 0.9029 - specificity_at_sensitivity: 0.9217 - rec_0: 0.3450 - pre_0: 0.5832 - rec_1: 0.9702 - pre_1: 0.9246 - macro_F1: 0.6902 - we
1000/1000 [==============================] - ETA: 0s - loss: 1.3216 - categorical_accuracy: 0.9029 - specificity_at_sensitivity: 0.9217 - rec_0: 0.3451 - pre_0: 0.5835 - rec_1: 0.9703 - pre_1: 0.9247 - macro_F1: 0.6903 - we
ighted_F1: 0.8916
```

Figure 6. Metrics results at the end of the first epoch



```
 975/1000 [============================>.] - ETA: 1s - loss: 0.0614 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8741 - pre_0: 0.9722 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9572 - we
 976/1000 [============================>.] - ETA: 1s - loss: 0.0614 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8742 - pre_0: 0.9722 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9573 - we
 977/1000 [============================>.] - ETA: 1s - loss: 0.0613 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8743 - pre_0: 0.9722 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9573 - we
 978/1000 [============================>.] - ETA: 1s - loss: 0.0613 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8744 - pre_0: 0.9723 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9574 - we
 979/1000 [============================>.] - ETA: 1s - loss: 0.0612 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8746 - pre_0: 0.9723 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9574 - we
 980/1000 [============================>.] - ETA: 1s - loss: 0.0612 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8748 - pre_0: 0.9724 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9575 - we
 981/1000 [============================>.] - ETA: 1s - loss: 0.0612 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8748 - pre_0: 0.9724 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9575 - we
 982/1000 [============================>.] - ETA: 1s - loss: 0.0611 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8749 - pre_0: 0.9725 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9575 - we
 983/1000 [============================>.] - ETA: 0s - loss: 0.0611 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8750 - pre_0: 0.9723 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 984/1000 [============================>.] - ETA: 0s - loss: 0.0611 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8750 - pre_0: 0.9724 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 985/1000 [============================>.] - ETA: 0s - loss: 0.0613 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8750 - pre_0: 0.9723 - rec_1: 0.9980 - pre_1: 0.9899 - macro_F1: 0.9575 - we
 987/1000 [============================>.] - ETA: 0s - loss: 0.0614 - categorical_accuracy: 0.9887 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8750 - pre_0: 0.9719 - rec_1: 0.9979 - pre_1: 0.9900 - macro_F1: 0.9574 - we
 989/1000 [============================>.] - ETA: 0s - loss: 0.0613 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8751 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9574 - we
 991/1000 [============================>.] - ETA: 0s - loss: 0.0611 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8752 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 993/1000 [============================>.] - ETA: 0s - loss: 0.0610 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8752 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 994/1000 [============================>.] - ETA: 0s - loss: 0.0610 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8752 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 995/1000 [============================>.] - ETA: 0s - loss: 0.0609 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8752 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 996/1000 [============================>.] - ETA: 0s - loss: 0.0609 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8753 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9900 - macro_F1: 0.9575 - we
 997/1000 [============================>.] - ETA: 0s - loss: 0.0608 - categorical_accuracy: 0.9888 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8753 - pre_0: 0.9719 - rec_1: 0.9980 - pre_1: 0.9901 - macro_F1: 0.9575 - we
 998/1000 [============================>.] - ETA: 0s - loss: 0.0608 - categorical_accuracy: 0.9889 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8753 - pre_0: 0.9720 - rec_1: 0.9980 - pre_1: 0.9901 - macro_F1: 0.9576 - we
 999/1000 [============================>.] - ETA: 0s - loss: 0.0607 - categorical_accuracy: 0.9889 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8753 - pre_0: 0.9720 - rec_1: 0.9980 - pre_1: 0.9901 - macro_F1: 0.9576 - we
1000/1000 [==============================] - ETA: 0s - loss: 0.0607 - categorical_accuracy: 0.9889 - specificity_at_sensitivity: 0.9972 - rec_0: 0.8755 - pre_0: 0.9720 - rec_1: 0.9980 - pre_1: 0.9901 - macro_F1: 0.9576 - we
ighted_F1: 0.9886
```

Figure 7. Metrics results at the end of the tenth epoch

- **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.

Loss	Categorical Accuracy	Specificity at sensitivity	Recall 0	Precision 0	Recall 1	Precision 1	Macro F1	Weighted F1
1.3216	0.9029	0.9217	0.3451	0.5835	0.9703	0.9247	0.6903	0.8916
0.1987	0.9772	0.9898	0.6719	0.9284	0.9967	0.9794	0.8838	0.9755
0.1249	0.9791	0.9946	0.7895	0.9576	0.9968	0.9807	0.9271	0.9782
0.1254	0.9710	0.9960	0.7913	0.9515	0.9947	0.9731	0.9239	0.9698
0.1123	0.9770	0.9957	0.7466	0.9524	0.9970	0.9786	0.9092	0.9757
0.0838	0.9863	0.9962	0.8095	0.9562	0.9976	0.9879	0.9347	0.9857
0.0963	0.9824	0.9961	0.8280	0.9666	0.9973	0.9837	0.9412	0.9818
0.0943	0.9808	0.9963	0.8294	0.9690	0.9971	0.9819	0.9416	0.9802
0.0737	0.9858	0.9970	0.8873	0.9731	0.9972	0.9871	0.9602	0.9855
0.0607	**0.9889**	**0.9972**	**0.8755**	**0.9720**	**0.9980**	**0.9901**	**0.9576**	**0.9886**

As we can see from Table 1, the metrics used to evaluate our model produced a great performance, since every measure at the end of the tenth epoch (last line of the table) was near 99%, except for the Recall 0 column (recall metric for the DoS label), which reached its best value at the ninth epoch (88.73%). It is important, however, to realize how the metrics evolve over time, generating better results for each step of the epochs (comparison between Figure 6 and 7). We expect, therefore, to achieve stronger outcomes as we increase the variable numbers that control the times the model is computed, which should also improve the Recall 0 metric. The variables' values used in this work were chosen considering its testing purpose only, but also to avoid getting an over-fitting scenario, since the recall and precision statistics were over 99%.

The reason why the Recall 0 metric presented a lower value over the training process can be explained by the fact that these scenarios mostly have unpredictable characteristics. Therefore, the high variability of the features may have complicated the generalization task of the model. It is also worth pointing out that the scenarios from the dataset were greatly unbalanced, with 70% of the cases being a benign flow and only 30% of the cases consisting of DoS attacks. This may be the biggest cause that prevented the model from find a good approximation for the malign flow scenarios.

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 researches 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 dataset and the results achieved confirmed the remarks pointed out by the recent studies. For future works, we intend to improve the study using simulation tools, such as NS3, SUMO and Omnet++, but also to experiment with novel GNN algorithms by applying newly reviewed concepts in other works.

References

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

Ammar, M., Russello, G., and Crispo, B. (2018). Internet of things: A survey on the security of iot frameworks. *Journal of Information Security and Applications*, 38.

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.

Baccari, S., Touati, H., Hadded, M., and Muhlethaler, P. (2020). Performance impact analysis of security attacks on cross-layer routing protocols in vehicular ad hoc networks. *2020 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*.

Busch, J., Kocheturov, A., Tresp, V., and Seidl, T. (2021). Nf-gnn: Network flow graph neural networks for malware detection and classification. *33rd International Conference on Scientific and Statistical Database Management (SSDBM 2021)*.

Chase, J. (2013). The evolution of the internet of things. *Texas Instruments*.

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*.

Cook, A. A., Mısırlı, G., and Fan, Z. (2020). Anomaly detection for iot time-series data: A survey. *IEEE Internet of Things Journal*, 7:6481 – 6494.

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*.

Deng, Y., Zhang, T., Lou, G., Zheng, X., Jin, J., and Han, Q.-L. (2021). Deep learning-based autonomous driving systems: A survey of attacks and defenses. *IEEE Transactions on Industrial Informatics*, 17:7897 – 7912.

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

Gonçalves, F., Ribeiro, B., Gama, O., Santos, J., Costa, A., Dias, B., Nicolau, M. J., Macedo, J., and Santos, A. (2020a). Dataset collection. `https://github.com/fabio-r-goncalves/dataset-collection`. [Online; accessed June 2024].

Gonçalves, F., Ribeiro, B., Gama, O., Santos, J., Costa, A., Dias, B., Nicolau, M. J., Macedo, J., and Santos, A. (2020b). Synthesizing datasets with security threats for vehicular ad-hoc networks. *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*.

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.

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.*

ITS, E. T. (2019). Cooperative intelligent transport systems (c-its) guidelines on the usage of standards. *ETSI European Standard*.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Springer Open Access*.

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.

Liu, L., Lu, S., Zhong, R., Wu, B., Yao, Y., Zhang, Q., and Shi, W. (2021). Computing systems for autonomous driving: State-of-the-art and challenges. *IEEE Internet of Things Journal*, 8:6469 – 6486.

Malik, S., Khan, M. A., El-Sayed, H., Khan, J., and Ullah, O. (2022). How do autonomous vehicles decide? *Sensors (ISSN 1424-8220)*, 23.

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., Jaekel, A., 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 - Published by Springer*, 16:2153 – 2185.

Pujol-Perich, D., Suárez-Varela, J., Cabellos-Aparicio, A., and Barlet-Ros, P. (2021). 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.*

Tang, Y., Zhang, C., Gu, R., and Li, P. (2015). Vehicle detection and recognition for intelligent traffic surveillance system. *Multimedia Tools and Applications - Springer*, 76:5817 – 5832.

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

Wu, L., Lei, S., Liao, F., Zheng, Y., Liu, Y., Fu, W., Song, H., and Zhou, J. (2024). Egconmix: An intrusion detection method based on graph contrastive learning. *arXiv e-prints.*

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).*