AutoMHS-GPT: Automated Model and Hyperparameter Selection with Generative Pre-Trained Model

Lucas Airam Castro de Souza, Matteo Sammarco, Nadjib Achir, Miguel Elias Mitre Campista, Luís Henrique Maciel Kosmalski Costa

AutoMHS-GPT: Automated Model and Hyperparameter Selection with Generative Pre-Trained Model

Lucas Airam C. de Souza[1,2], Matteo Sammarco[3], Nadjib Achir[2],
Miguel Elias M. Campista[1], Luís Henrique M. K. Costa[1]

Grupo de Teleinformática e Automação (GTA) Universidade Federal do Rio de Janeiro (UFRJ)
[2]École Polytechnique, INRIA Saclay, France
[3] Stellantis

***Abstract*—Automated Machine Learning emerges as a solution to reduce the instantiation time of systems that rely on Artificial Intelligence (AI) by accelerating the search process for models and hyperparameters. These techniques, however, still require high execution time. In critical applications, such as intrusion detection in vehicular networks, delays in applying countermeasures can provoke accidents. Therefore, it is essential to guarantee accurate models in the shortest possible time to detect threats effectively. This work proposes AutoMHS-GPT, a system that uses generative artificial intelligence to reduce the time it takes to define hyperparameters and models when implementing machine learning to detect threats in vehicular networks. Based on a description of the problem, the generative model returns a text containing the appropriate model with its hyperparameters for training. Results show that AutoMSH-GPT produces models with higher threat classification performance than automated machine learning approaches AutoKeras and Auto-Sklearn, increasing in the best case the recall by 9%. Furthermore, the current proposal reduces the model search and training process, carrying out the task in around 30 minutes, while the other evaluated frameworks require two to three days.**

***Index Terms*—Machine Learning; Generative AI; Network Security; Vehicular Networks; AutoML.**

I. Introduction

The capability to automate tasks and reduce costs makes machine learning an attractive technology for several areas, such as computer networks, intelligent vehicles, medicine, and others. However, deploying artificial intelligence models in production environments takes days or even months [1]. This delay occurs due to the data collection and model training process. A model requires training time in the order of minutes [2]. Execution time varies depending on the dataset size, the chosen model, and available computational resources. However, with the wide availability of existing models and hyperparameters, it is difficult to determine model hyperparameters to use for a new learning task. Thus, a common approach is to apply hyperparameter optimization.

Optimizing models and hyperparameters is the most computationally intensive and time-consuming task since search processes are generally in large dimensional spaces. Therefore, automating models and hyperparameters definition is essential for reducing the time it takes to make a machine learning model available. Furthermore, automating machine learning steps allows users with little technical knowledge to develop high-quality machine learning models, only worrying about the data acquisition process.

Thus, Automated Machine Learning (*Automated Machine Learning* - AutoML) [3], [4], [5] is a research field that seeks to automate the learning process, where the combined selection of algorithm and optimization of hyperparameters (Combined Algorithm Selection and Hyperparameter Optimization - CASH) is one of the specific cases addressed by AutoML [6]. AutoML uses Bayesian optimization techniques to reduce the search space and determine the most promising hyperparameters and models, generally considering a maximum execution time. However, these techniques still require high execution time or produce results with low classification performance when the search time is short. The scenario becomes even worse in vehicular networks, where applications are sensitive to latency, particularly in intrusion detection systems, where a vulnerable vehicle can be compromised by an attacker, putting the physical safety of passengers at risk.

We propose AutoMHS-GPT (Automated Model and Hyperparameter Selection with Generative Pre-Trained model)[1], a system that uses generative artificial intelligence to reduce the search time for hyperparameters and models in the instantiation of machine learning services. We consider in our proposal the intrusion detection in vehicular networks learning task as a use case and ChatGPT [7] is used as generative artificial intelligence (*Generative Artificial Intelligence* - GenAI). From a description of the problem, "determine the best model and set of hyperparameters for a vehicular networks attacks dataset which contains tabular data", GenAI returns a text containing an appropriate answer. After this step, training

[1]A preliminary version of this paper was published in Portuguese and is available at https://www.gta.ufrj.br/ftp/gta/TechReports/SSA24.pdf.

begins with the result transformed into code that implements the model. Results show that AutoMSH-GPT produces models with higher threat classification performance than evaluated automated machine learning approaches AutoKeras and Auto-Sklearn, delivering 9% higher recall in the best case. The same effect occurred in other metrics, such as accuracy, precision, and F1 score, demonstrating that the model correctly classifies more samples and generates fewer false positives than the alternatives evaluated. Furthermore, the current proposal reduces the model search and training process, carrying out the task in around 30 minutes, while the other evaluated frameworks require two and three days to determine the model.

We organize the paper as follows. Section II presents the state of the art in intrusion detection in vehicular networks and applications of generative learning in computer networks. Section III describes the proposed system, the steps, and the data used for model selection, hyperparameters, and subsequent training. Section IV evaluates the system and compares it with other state-of-the-art AutoML proposals: AutoKeras and Auto-Sklearn. Finally, Section V concludes this work and presents future research directions.

II. Related Works

This section presents state-of-the-art solutions for threat detection in vehicular networks and the use case adopted by the proposed system. Furthermore, we discuss alternatives for hyperparameter optimization and model selection in machine learning systems. Finally, we exhibit applications of generative models in computer networks. The discussion demonstrates that the main applications focus on detecting anomalies or generating new samples for model training.

A. Threat Detection in Intelligent Vehicle Networks

Intrusion detection in vehicular environments is vital to ensure the safety of drivers and passengers. Unlike other computer systems, a compromised vehicle by an attacker can cause accidents, including fatalities. Yakan *et al.* propose an intrusion detection system for vehicular networks focusing on Vehicle-to-Network (*Vehicular-to-Network* - V2N) [8] communication. The authors use *Long Short-Term Memory* (LSTM) models to capture temporal relationships in attacks and Federated Learning (*Federated Learning* - FL) to increase privacy and communication efficiency during training. Vinita and Vetriselvi [9] propose a system to identify the integrity of emergency messages transmitted over vehicular networks.

On the one hand, Vehicle-to-Everything (V2X) communication brings opportunities to increase driving efficiency by sending an update on local traffic conditions, such as reporting accidents. On the other hand, attackers can spread false messages to degrade traffic conditions. Therefore, the authors propose using machine learning models to detect false accident reports through Sybil attacks. Furthermore, the proposal uses the federated learning paradigm to preserve users' privacy. Bousalem *et al.* [10] propose using reinforcement learning to mitigate DDoS attacks in vehicular networks. The 5G-V2X network allows you to instantiate slices with few communication resources to isolate attackers or clients with suspicious behavior. However, these isolated nodes must recover resources after the threat has ceased. Thus, the authors propose a reinforcement learning algorithm to determine when to reduce users' resources by assigning them to a resource-limited slice and when to increase their resources again. Although the proposals are promising in combating threats in vehicular networks, the procedures developed to select models and adjust their hyperparameters represent an open challenge.

B. Model Selection and Hyperparameter Optimization

Model selection and hyperparameter optimization are two of the main challenges that exist in the development of machine learning systems. Thus, a relevant research topic is hyperparameter optimization [11], [12]. AutoML [3], [4], [5] is a research area focused on automating the learning process, defining data pre-processing, models, and hyperparameters. CASH is one of the specific cases of AutoML in which the objective is to select models and hyperparameters [6]. There are several AutoML implementations available, such as Auto-Sklearn [3], H2O [5], AutoKeras [4], and Google AutoML [13]. Furthermore, there are other proposals such as Auto-CASH [14] and the proposal by Horváth *et al.* [12]. Auto-CASH is a tool that uses reinforcement learning to automate the hyperparameter and model selection process and thereby reduce the need for human intervention. In other ways, Horváth *et al.* use Principal Component Analysis (*Principal Component Analysis* - PCA) in conjunction with a similarity measure as a way of creating meta-features that assist the process of making decisions in the same optimization task. However, these techniques still require a long execution time to achieve satisfactory results. However, the increasing development of applications that use generative models raises the question of the ability to solve this problem faster than current techniques while also providing accurate models.

C. Generative Models in Computer Networks

The use of generative artificial intelligence (*Generative AI* - GenAI) on computer networks is a research topic currently explored. Jacobs *et al.* [15] propose using a prompt for automatic network configuration, analyzing users' intentions through a high-level description. Zhang *et al.* [16] study the use cases of generative AI in vehicular networks. The authors identify three vehicular network applications that can be improved using generative AI: traffic simulation, data augmentation, and risk assessment. Furthermore, the authors propose reducing communication with high-fidelity recreation of sending accident information. The proposal is to send the most relevant image characteristics with descriptive text to reconstruct the original in other vehicles.

Due to the ability of these models to generate new data from previous observations, generative AI is used to augment

datasets in order to train other models on a larger set of samples. TrajGAIL (*Trajectory Generative Adversarial Imitation Learning*) [17] is a framework that applies generative AI to generate urban vehicle trajectories. The authors propose to combine a Partially Observable Markov Decision Process (*Partially Observable Markov Decision Process* - POMDP) with the generative model to generate more realistic data. Obtaining data is difficult due to challenges such as scarcity and privacy issues to collect datasets of urban trajectories, as well as the high computational cost of approaches such as Inverse Reinforcement Learning (*Inverse Reinforcement Learning* - IRL). Similarly, SCAN-GAN (*Synthetic Controller Area Network-Generative Adversarial Network*) [18] is a method for generating synthetic data, but for intrusion detection in vehicular networks.

Another possibility for applying generative networks is the transformation of input characteristics to identify anomalies. Cobilean *et al.* propose a transformation-based anomaly detection system for CANs [19]. Since transformer networks detect dependencies in sequences, the authors use this feature to predict the likely value in a sequence of CAN messages. When the received value differs greatly from the forecast, the system detects an anomaly. Zhao *et al.* propose and evaluate four structures of an intrusion detection system on the CAN bus [20]. The authors combine a Generative Adversarial Network (*Generative Adversarial Networks*- GAN) with out-of-distribution detection (*Out-of-Distribution - OOD*) to classify data as normal, known attacks or unknown attacks. The article trains the GAN discriminator to distinguish sequences of benign and malicious messages, differently than previous proposals that use GAN as a data generator. Furthermore, the proposal applies an isolation forest model to detect unknown attacks. Du *et al.* [21] present applications of generative AI in computer networks and discuss how this technology can impact security. Generative AI can be a source of malicious code or data to poison discriminative models such as machine learning-based intrusion detection systems. However, this technology is also capable of extracting important characteristics of attacks or even creating new attack patterns to train other models.

Gupta *et al.* [22] present the applications for security in computer networks using ChatGPT (Generative Pre-trained Model - *Generative Pretrained Model*). The authors discuss the ability of this tool to generate codes to execute attacks or countermeasures. Despite hallucination problems, the model generates codes that perform the tasks required in different scenarios tested by the authors. Juttner *et al.* [23] use ChatGPT as a mechanism to explain to non-expert users the result of the classification performed by Intrusion Detection Systems (*Intrusion Detection Systems* - IDS). Furthermore, the authors propose to use the generative model to indicate necessary countermeasures.

Unlike previous proposals, we use generative artificial intelligence to produce a machine learning model with hyperparameters adjusted to the learning task based on information about the data set and the learning problem. The proposal allows for reducing model generation time, searching for suitable options for training, and eliminating complex tasks such as optimization through exhaustive searches for hyperparameters and models. On the other hand, the proposal is dependent on the generative model without guaranteeing that the generated model is optimal. Furthermore, the work addresses the scenario of threat detection in vehicular networks as its main application, although it is possible to adapt it to other learning problems.

III. The System AutoMHS-GPT

AutoMHS-GPT consists of a system for defining models and hyperparameters using artificial intelligence. The system has an intelligence service that uses generative artificial intelligence to estimate the best hyperparameters, including the learning model, based on information about the data set. The information considered is the types of characteristics of the dataset and the learning task. Thus, the generative model receives as input a sentence in natural language that contains the predicted information and generates as a response a model with the appropriate hyperparameters for the classification problem. Therefore, the proposal applies the pre-trained generative model as a form of AutoML. The response time, however, is less than necessary to obtain a classification model compared to tools such as AutoKeras and Auto-Sklearn. The proposed use case consists of training models to identify threats in vehicular networks.

The execution of the system starts with a user's request to the intelligence service through the vehicle safety module. AutoMHS-GPT uses ChatGPT for hyperparameter and model selection. Therefore, the system receives as input a sentence in natural language "I have a classification problem that uses the VeReMi dataset and I want to know which is the best model with the defined hyperparameters for this task. The dataset is tabular and I only wish to receive a model and its hyperparameters.". After processing this sentence, the ChatGPT returns a machine learning model with the configured hyperparameters. Hyperparameters missing after this step are set to the default value of the machine learning library used. Because the generative model is pre-trained, the response is generated in a few seconds. The main advantage of the proposal is the definition of the learning model and its hyperparameters before training. Furthermore, the generative model is adaptable to other classification tasks or datasets by changing the input sentence. On the other hand, alternatives for hyperparameter and model optimization run the training process multiple times to determine the best model. Although the search has heuristics to reduce execution time, a change in the learning problem requires the execution of several models again.

A limitation of the current AutoMHS-GPT implementation is the dependency on ChatGPT. Although the text used as input and the *prompt* are adjustable, the generative model is used as an external application programming interface

(*Application Programming Interface* - API). Therefore, the system does not control the data used to train the generative model at first. On the other hand, AutoML approaches spend more time searching for the model with the appropriate hyperparameters for the classification task. In the case of applications with greater tolerance to high model search time and use of computational capacity, AutoML is capable of evaluating more possibilities about the data set and finding a better model. Therefore, there is a trade-off in response time and performance. Depending on the application, response time is predominant, as in a dynamic environment such as the vehicle addressed by the current proposal.

Figure 1 displays the architecture of the proposed system. White modules and components are features not implemented by the current work. The system architecture contains six services, three of which are presented in this paper. The (1) vehicle safety service manages the learning models to be executed in the vehicle. This service has two modules, (a) model request, responsible for searching for models for threat detection, and module (b) responsible for executing the model in the vehicle. Module (a) executes in the cloud, while threat detection executes in the vehicle. The (2) intelligence service receives requests for models and, based on the received information, searches for an appropriate model for the classification problem. If no model is found, the model selection module starts training a new model. This service applies generative artificial intelligence to analyze requests and return a suitable model. The (3) machine learning model training service is responsible for adjusting the parameters of the models defined by the intelligence service. The trained models are stored in a model database to be accessed and used later. Currently, the system has a centralized training module, which can be used for learning tasks in which the information transmitted is not sensitive and can be stored in the cloud. The remaining modules and services will be implemented in future work.

IV. Prototype Development and Results Obtained

This section presents the data set used to evaluate the created models, characterizes the execution environment, and presents the results obtained.

A. Dataset Description

Heijden *et al.* [24] present the publicly available *Vehicular Reference Misbehavior Dataset* (VeReMi) and evaluate plausibility mechanisms in the generated data. The dataset consists of message records for each vehicle in the simulation and a basic information file that specifies the attackers' behavior. The messages are of the CAM type. VeReMi contains five attacks: constant position, constant displacement, random position, random displacement, and eventual stop. All attacks are related to the car's position, sending either the same location or the actual position plus a noise signal, which can be constant or random. The eventual stop attack simulates a byzantine behavior, where the car starts sending the real position, but suddenly starts sending a constant false position. Kamel *et al.* [25] extend the previous dataset to include more data and attack patterns. Thus, the VeReMi extension contains patterns such as delayed messages, DoS, data replay, fake vehicle message diffusion, and speed malfunction, in addition to the position malfunction attacks in the first version of the dataset, described below.

The attacker's main objective is to cause disturbance in the vehicular environment, either by unavailability of communication or by sending false data. The unavailability of communication reduces the advantages obtained by cooperative driving systems, such as information on traffic conditions in a region. Likewise, sending false data harms vehicles' decisions about routes between origin and destination. For example, the information that a road has many cars, or that the vehicle's speed is low can be a decisive factor for other vehicles to follow a different route. Furthermore, the attacker can cause accidents by reporting a sudden stop to nearby vehicles and forcing them to stop unnecessarily or perform a lane change.

There are 19 types of attacks in the VeReMi Extension dataset. Constant position (1) the vehicle with malicious behavior reports the same fixed position despite its movement. Constant position offset (2) the vehicle adds a predetermined fixed value to its current position, resulting in the generation of a path parallel to the true path. Random position (3) the vehicle sends a random position instead of the true position. Random position shift (4) the vehicle adds a random number to its true position, creating a confusing path. The constant speed (5), constant speed change (6), random speed (7), and random speed change (8) attacks are respectively similar to the first four, the only difference is that the attacker changes his speed value instead of its location. Eventual stop (9) the vehicle initially sends its precise location and speed information, but after a certain time, it begins to report a zero position and speed. Disruptive (10) the vehicle reproduces information previously received from random neighbors to overload the network. Data replay (11) the vehicle reproduces the information received from a specific given neighbor as if it were its own.

Other attacks directly affect communication, such as delayed messages (12) in which the attacking vehicle sends precise information about its previously recorded movement after a pre-defined time. DoS (13) the attack consists of flooding the network with information to make other vehicles unavailable. Random DoS (14) attack is similar to DoS, however, all values included in the messages are random and inaccurate. Disruptive DoS (15) combines increasing the sending frequency with the flooding of previously received random messages. Traffic congestion Sybil (16) the attacker intends to create a false congestion reporting the existence of ghost vehicles, where pseudo-vehicle IDs are created for non-existent vehicles in a specific target position, and the attacker maintains a realistic communication position with ghost vehicles. Sybil data replay (17) is similar to attack 11, but more sophisticated. The attacker replays the data

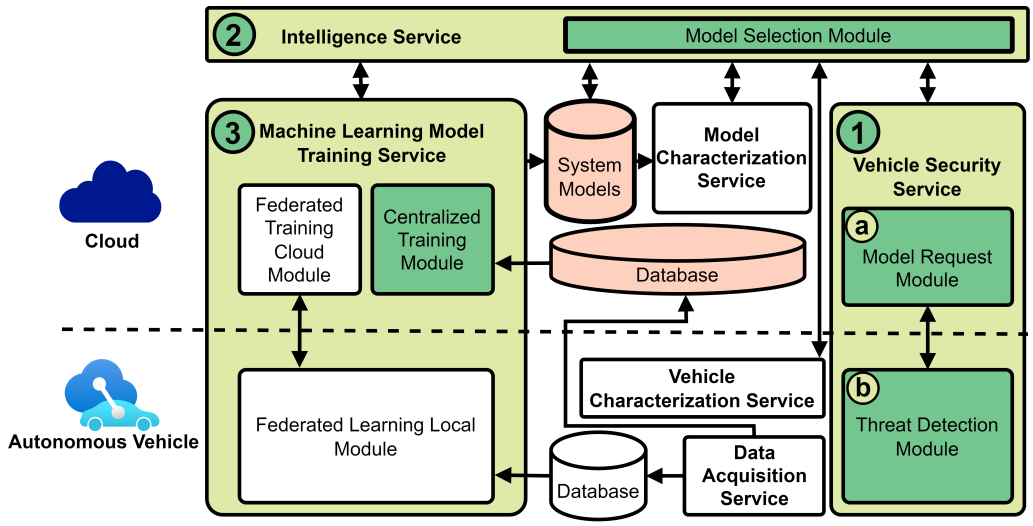

Figure 1. AutoMHS-GPT architecture. This work implements the intelligence service, security service, and centralized training.

of a target neighbor using multiple pseudo-IDs to mask the identity of the real attacker. Random Sybil DoS (18) combines three forms of attacks into one: increased frequency of sent messages, randomization of all values in messages, and multiple fake pseudo-IDs. Disruptive Sybil DoS (19) the attacker replays previously received messages from neighbors randomly with high frequency using many pseudo IDs.

B. Test environment

A prototype of the proposed system was implemented in Python v3.10.12 and tested on models created with the scikit-learn library v1.2.1[2]. Furthermore, generative artificial intelligence is implemented by ChatGPT [7], with the AutoML frameworks AutoKeras [4] v1.1.0 and Auto-Sklearn [3] v0.15.0 used for comparison. The two selected frameworks stand out for being open-source and easy to use, an advantage compared to other AutoML frameworks. The experiments were carried out on an Intel Xeon E5-2650 CPU 2.00 GHz server with 32 processing cores and 504 GB of RAM. Finally, the results obtained are demonstrated with 95% confidence intervals. The dataset was partitioned with 80% of the data for training and 20% for testing. The data set is unbalanced, with the normal class being the majority. This configuration was maintained in the experiments, with the only pre-processing performed being the elimination of highly correlated features based on Pearson correlation.

AutoKeras and Auto-Sklearn require parameter definition before starting the search process for models and hyperparameters. Thus, Table I displays the configuration of parameters used for AutoML frameworks. AutoKeras requires the number of epochs for training each model and the total number of models to be evaluated. Auto-Sklearn requires the maximum time to execute each search task and the total search time. Furthermore, both proposals have a model used to optimize hyperparameters. The choice of values for these AutoML parameters is discussed in the execution time comparison of each approach. On the other hand, AutoMHS-GPT uses ChatGPT as a search engine for the same activity. After processing the text in natural language, the system returned a random forest model, whose hyperparameters are 150 trees, two samples as the minimum number to divide, and one sample at least per leaf. The remaining hyperparameters remained with the default configuration. As the generative model is pre-trained, the response is generated in a few seconds. The classification performance results using the different approaches are presented below.

[2]Scikit-learn [26] is an open source, well-documented library, for creating machine learning models, which has numerous developers. Available at https://scikit-learn.org/

Table I
PARAMETERS USED FOR AUTOML FRAMEWORKS.

Framework	Paramter	Value
AutoKeras	Model	StructuredDataClassifier
	Total Attempts	30
	Total Training Epochs	100
Auto-Sklearn	Model	AutoSklearnClassifier
	Maximum Time	1725 minutes
	Timeout per Attempt	573 minutes

C. Comparison between AutoMHS-GPT and State of the Art

This section presents the experiments to evaluate and compare AutoMHS-GPT with the state-of-the-art. The first part evaluates the classification performance, while the second experimental part evaluates the time required to execute the proposals until the models are instantiated.

1) Classification Performance: The first experiment carried out evaluates the classification performance of the different proposals. As the dataset has 20 distinct classes, it is possible to use two approaches, binary or multiple-class

classification. Binary classification consists of determining whether the transmitted message is part of an attack or normal communication. However, this type of classification reduces the granularity in the countermeasure instantiation process, as the detection system only knows the existence of the attack. Therefore, the results consider the classification of multiple classes for specific reactions to each type of attack later.

VeReMi Dataset: First, we evaluate the performance of three frameworks on VeReMi dataset, with six different classes. Figure 2 displays the accuracy of the three approaches. AutoMHS-GPT classifies more samples correctly than the other proposals, presenting 0.62% more accuracy than the model generated by Auto-Sklearn and 11.48% higher than AutoKeras.

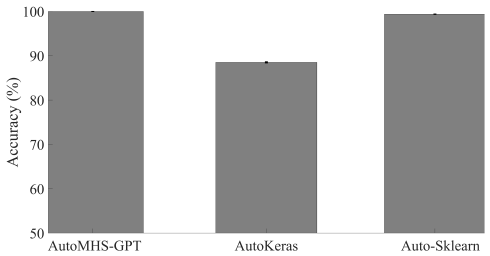

Figure 2. Accuracy evaluation of the three proposals on the VeReMi dataset.

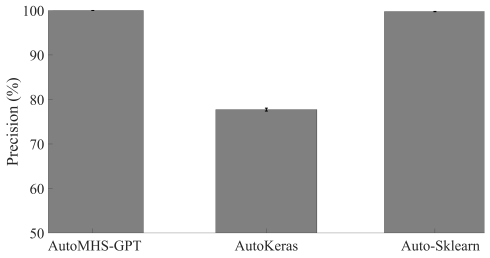

Figure 3. Preicision evaluation of the three proposals on the VeReMi dataset.

We also observe that the precision of AutoMHS-GPT's model is higher than the other two approaches. However, Auto-Sklearn has a precision value close to AutoMHS-GPT, 99.76% and 99.99% respectively. This means that the model generated by AutoMSH-GPT produces fewer false positives than Auto-Sklearn. Also, both approaches have better results than AutoKeras, which has only 77.73% of precision, generating a higher amount of false positives. Figure 3 presents this result.

Regarding the recall, the model generated by AutoKeras is capable of detecting only 71% of attacks. This means that around 29% can affect the system when deploying a model trained by AutoKeras with VeReMi dataset. Moreover, AutoMHS-GPT and Auto-Sklearn detect 99.94% and 97.97% of attacks, respectively, as shown in Figure 4.

Finally, for the this first dataset, we evaluate the F1-score and display the results in Figure 5. This metric is a harmonic

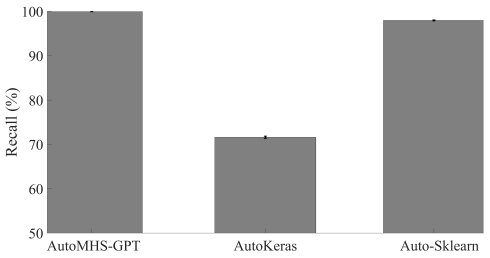

Figure 4. Recall evaluation of the three proposals on the VeReMi dataset.

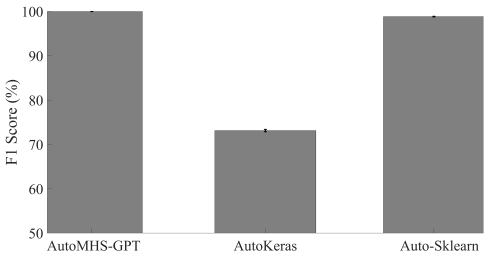

Figure 5. F1 score evaluation of the three proposals on the VeReMi dataset.

mean between the precision and recall. Since the metrics follow the same pattern and similar values, the harmonic maintains almost the same result as the other two metrics.

AutoKeras has the lowest values for all classification metrics for the VeReMi dataset. One possible solution to increase its classification performance is to change the AutoML parameters, e.g. increasing the number of epochs. However, this approach will also increase the time to search for models, which we reduce with our proposal. Then, a better solution is to select Auto-Sklearn or AutoMHS-GPT as the AutoML strategy for the VeReMi dataset. We execute the same experiment on the VeReMi Extension dataset to validate these first results. The VeReMi Extension dataset is a more complex problem than the first dataset, because it has more classes and more features.

VeReMi Extention Dataset: At the second part of the first experiments, we evaluate the performance of three frameworks on VeReMi Extension dataset, with 20 different classes. This is a more complex problem compared to the first dataset, because it has more classes and more features. Figure 6 displays the accuracy of the three approaches evaluated on the attack dataset on vehicular networks. AutoMHS-GPT classifies more samples correctly than the other proposals, presenting 1.35% better performance compared to the model generated by Auto-Sklearn and 3.99% better compared to AutoKeras.

The same behavior can be observed concerning accuracy, where the model generated by the current proposal presents an average accuracy of 90.74%, while the others generated models with 79.44% and 89.65%, for AutoKeras and Auto-Sklearn, respectively. Figure 7 presents this result.

Furthermore, AutoMHS-GPT increases the ability to detect

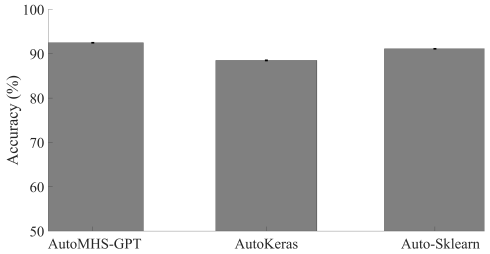

Figure 6. Accuracy evaluation of the three proposals on the VeReMi Extention dataset.

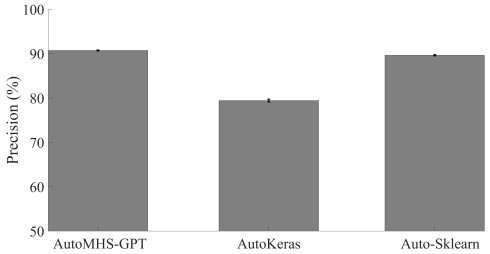

Figure 7. Precision evaluation of the three proposals on the VeReMi Extention dataset.

attacks, as it has a higher recall than other proposals. The result is shown in Figure 8. Finally, the F1 Score of the proposed system is also higher than the others, since both precision and recall are higher than those of the compared frameworks, as shown in Figure 9.

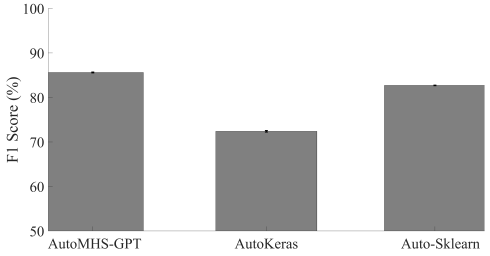

Figure 9. F1 score evaluation of the three proposals on the VeReMi Extention dataset.

The generative artificial intelligence used as a model and hyperparameter search engine in the threat detection scenario in vehicular networks has a superior performance than AutoML. Another important evaluation metric for the current problem is execution time. Therefore, the second part of the experiment consists of evaluating this metric for each approach.

D. Model Training Performance

The execution time comprises model definition, hyperparameters, and training time. We execute this experiments for both datasets, because they have different sample spaces and feature spaces. Thus, for VeReMi dataset, we expect that the

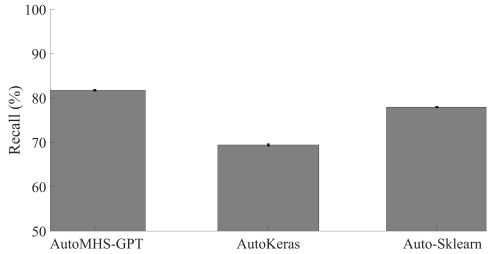

Figure 8. Recall evaluation of the three proposals on the VeReMi Extention dataset.

execution time will be shorter than the same metric on the VeReMi Extension.

We exhibit the results for the VeReMi dataset in Figure 10. AutoMHS-GPT has an execution time of approximately two orders of magnitude shorter than other AutoML tools. This occurs because the process of searching for hyperparameters and models is carried out by the pre-trained model, obtaining the result in a few seconds. Therefore, the main factor in the execution time of the proposal is the training of the model resulting from the proposal for threat classification.

Figure 11 shows the result of this experiment for the second dataset. The tendency is the same as the results obtained for the first dataset. Comparing the results between the two datasets, the time to generate the models on the VeReMi dataset is slightly lower as expected, for all proposals. Since the dataset is smaller, each search executed by the AutoML is also faster than the search on VeReMi Extension. Nonetheless, on both datasets, the execution time is in order of minutes for the model generated with AutoMHS-GPT, while the AutoML frameworks need more than one day to search for the best model.

The execution time of AutoML frameworks is dependent on the parameters displayed in Table I. Reducing the number of epochs in the case of AutoKeras generates inaccurate models. Auto-Sklearn is more sensitive to adjusting the timeout per attempt. A very strict timeout makes the framework unable to train models, generating only a dumb classifier (*DummyClassifier*), which assigns all samples the majority class in the dataset. In the case of the evaluated data set, all samples are classified as normal by this classifier. Furthermore, the size of the dataset directly influences the execution time of all approaches. Therefore, for smaller datasets, the difference can be reduced. In the security scenario in vehicular networks, it is essential to generate a model in the shortest possible time. The absence of an updated model makes the vehicle vulnerable and susceptible to the attacks discussed above, putting the physical integrity of its passengers at risk. Also, the experiment demonstrates that our proposal reduces the model deployment time to approximately 30 minutes, while AutoML frameworks require approximately 3 days for the same task. Finally, the results show that Auto-Sklearn generates a model with a performance similar to AutoMHS-GPT

in both datasets. Nevertheless, the time to models' search of Auto-Sklearn is more than 41 times greater than the time needed for our proposal.

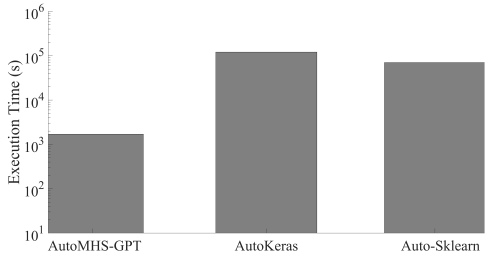

Figure 10. Proposals' execution time on the VeReMi dataset.

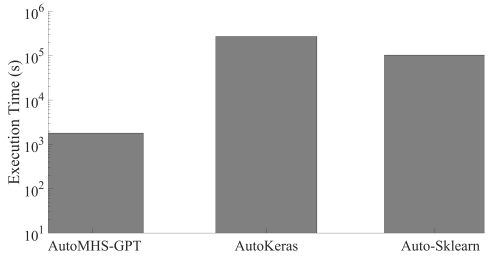

Figure 11. Proposals' execution time on the VeReMi Extention dataset.

V. Conclusion and Future Works

We presented AutoMSH-GPT, a system to automate model and hyperparameters definition based on information from the dataset. The automation of model selection is carried out through generative artificial intelligence based on data about the classification problem. The results show that AutoMSH-GPT generates a model with high classification performance and low time to define the model and its hyperparameters for training, compared to AutoML approaches. In future work, we intend to implement a transformer-based architecture that is specific to the task of model hyperparameter optimization and add new modules to the proposed system. We also envision training models through federated learning to preserve user privacy.

References

[1] A. Paleyes, R.-G. Urma, and N. D. Lawrence, "Challenges in Deploying Machine Learning: a Survey of Case Studies," *Computing Surveys*, vol. 55, no. 6, pp. 1–29, 2022.

[2] S. Kumar *et al.*, "Exploring the Limits of Concurrency in ML Training on Google TPUs," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 81–92, 2021.

[3] M. Feurer *et al.*, "Efficient and Robust Automated Machine Learning," *Advances in Neural Information Processing Systems (NIPS)*, vol. 28, 2015.

[4] H. Jin, Q. Song, and X. Hu, "Auto-Keras: An Efficient Neural Architecture Search System," in *International Conference on Knowledge Discovery & Data Mining (SIGKDD)*. ACM, 2019, pp. 1946–1956.

[5] E. LeDell and S. Poirier, "H2O AutoML: Scalable Automatic Machine Learning," in *AutoML Workshop (ICML)*. International Machine Learning Society, 2020.

[6] C. Thornton *et al.*, "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms," in *International Conference on Knowledge Discovery and Data Mining (SIGKDD)*. ACM, 2013, pp. 847–855.

[7] OpenAI, "ChatGPT: Optimizing Language Models for Dialogue," Available at: https://openai.com/blog/chatgpt/, 2023, Último acesso: 28 de janeiro de 2024.

[8] H. Yakan, I. Fajjari, N. Aitsaadi, and C. Adjih, "Federated Learning for V2X Misbehavior Detection System in 5G Edge Networks," in *Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*. ACM, 2023, pp. 155–163.

[9] L. J. Vinita and V. Vetriselvi, "Federated Learning-based Misbehaviour Detection on an Emergency Message Dissemination Scenario for the 6G-enabled Internet of Vehicles," *Ad Hoc Networks*, vol. 144, p. 103153, 2023.

[10] B. Bousalem *et al.*, "DDoS Attacks Mitigation in 5G-V2X Networks: A Reinforcement Learning-Based Approach," in *International Conference on Network and Service Management (CNSM)*. IEEE, 2023, pp. 1–5.

[11] H. N. C. Neto, I. Dusparic, D. M. Mattos, and N. C. Fernande, "FedSA: Accelerating Intrusion Detection in Collaborative Environments with Federated Simulated Annealing," in *International Conference on Network Softwarization (NetSoft)*. IEEE, 2022, pp. 420–428.

[12] T. Horváth *et al.*, "Hyper-Parameter Initialization of Classification Algorithms using Dynamic Time Warping: A Perspective on PCA Meta-Features," *Applied Soft Computing*, 2023.

[13] E. Bisong and E. Bisong, "Google AutoML: Cloud Vision," *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*, pp. 581–598, 2019.

[14] T. Mu *et al.*, "Auto-CASH: A Meta-Learning Embedding Approach for Autonomous Classification Algorithm Selection," *Information Sciences*, vol. 591, pp. 344–364, 2022.

[15] A. S. Jacobs *et al.*, "Hey, LUMI! Using Natural Language for Intent-Based Network Management," in *Annual Technical Conference (ATC)*. USENIX, 2021, pp. 625–639.

[16] R. Zhang *et al.*, "Generative AI-enabled Vehicular Networks: Fundamentals, Framework, and Case Study," *arXiv preprint arXiv:2304.11098*, 2023.

[17] S. Choi, J. Kim, and H. Yeo, "TrajGAIL: Generating Urban Vehicle Trajectories using Generative Adversarial Imitation Learning," *Transportation Research Part C: Emerging Technologies*, vol. 128, p. 103091, 2021.

[18] A. Chougule, K. Agrawal, and V. Chamola, "SCAN-GAN: Generative Adversarial Network Based Synthetic Data Generation Technique for Controller Area Network," *Internet of Things Magazine*, vol. 6, no. 3, pp. 126–130, 2023.

[19] V. Cobilean *et al.*, "Anomaly Detection for In-Vehicle Communication Using Transformers," in *Industrial Electronics Society (IECON)*. IEEE, 2023, pp. 1–6.

[20] Q. Zhao, M. Chen, Z. Gu, S. Luan, H. Zeng, and S. Chakrabory, "CAN Bus Intrusion Detection Based on Auxiliary Classifier GAN and Out-of-distribution Detection," *Transactions on Embedded Computing Systems (TECS)*, vol. 21, no. 4, pp. 1–30, 2022.

[21] H. Du *et al.*, "Spear or Shield: Leveraging Generative AI to Tackle Security Threats of Intelligent Network Services," *arXiv preprint arXiv:2306.02384*, 2023.

[22] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, "From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy," *arXiv preprint arXiv:2307.00691*, 2023.

[23] V. Jüttner, M. Grimmer, and E. Buchmann, "ChatIDS: Explainable Cybersecurity Using Generative AI," *arXiv preprint arXiv:2306.14504*, 2023.

[24] R. W. Van Der Heijden, T. Lukaseder, and F. Kargl, "VeReMi: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *Security and Privacy in Communication Networks (SecureComm)*. Springer, 2018, pp. 318–337.

[25] J. Kamel, M. Wolf, R. W. Van Der Hei, A. Kaiser, P. Urien, and F. Kargl, "VeReMi Extension: A Dataset for Comparable Evaluation of Misbehavior Detection in VANETs," in *International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[26] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.