Distributed Machine Learning and Multi-Agent Systems for Enhanced Attack Detection and Resilience in IoT Networks

Gustavo Funchal[1][a], Tiago Pedrosa[1][b], Fernando de la Prieta[2][c] and Paulo Leitão[1][d]

[1]*Research Centre in Digitalization and Intelligent Robotics (CeDRI), Laboratório Associado para a Sustentabilidade e Tecnologia em Regiões de Montanha (SusTEC), Instituto Politécnico de Bragança, 5300-253 Bragança, Portugal*

[2]*BISITE Digital Innovation Hub, University of Salamanca, Edificio I+D+i, C/ Espejos s/n, 37007, Salamanca, Spain*

Keywords: Intrusion Detection Systems, Multi-Agent Systems, Internet of Things, Machine Learning.

Abstract: The exponential growth of connected devices, including sensors, mobile devices, and various Internet of Things (IoT) devices, has resulted in a substantial increase in data generation. Traditionally, data analysis involves transferring data to cloud computing systems, leading to latency issues and excessive network traffic. Edge computing emerges as a promising solution by bringing processing closer to the data sources. However, edge computing faces challenges, particularly in terms of limited computational power, which can create constraints in the execution of machine learning (ML) tasks. This paper aims to analyze strategies for distributing ML tasks among multiple nodes based on multi-agent systems (MAS) technology to have a collaborative approach and compare these strategies to provide an overview of best practices for achieving the optimal performance in intrusion detection for Industrial Internet of Things (IIoT). In this way, the well-known CICIoT2023 data set was used, and centralized and distributed ML techniques were implemented, and evaluated. The distributed edge ML approach achieved promising results, presenting an improvement of between 7.73% and 32.18% in the correction of wrong predictions of detection of attacks on IoT devices, significantly improving the precision and recall of the applied techniques.

1 INTRODUCTION

The rapid expansion of connected devices, encompassing sensors, mobile devices, and a myriad of Internet of Things (IoT) devices, has assisted in an era of unprecedented data generation. The conventional practice of transferring this vast amount of data to cloud systems for analysis has proven to be susceptible to latency issues and network congestion (Popescu et al., 2017). In response to these challenges, edge computing has emerged as a transformative paradigm (Tyagi and Tyagi, 2024), aiming to process data in proximity to its source.

In the context of Industry 4.0 (Kagermann et al., 2013), the integration of digital technologies into industrial processes, the reliance on connected devices and IoT solutions has become integral. Industry 4.0 promises increased efficiency, automation, and connectivity in manufacturing and industrial settings. However, this evolution has also given rise to new challenges, particularly in the realm of cybersecurity.

In recent times, there has been a rise in security issues within the Industry 4.0 landscape, with cyber threats becoming more sophisticated and widespread. The interconnected nature of devices and systems within the Industrial Internet of Things (IIoT) has exposed vulnerabilities that require strong security measures (Pourrahmani et al., 2023). As industries aim to benefit from digital transformation, ensuring the security of these interconnected systems becomes crucial.

One of the critical aspects of cybersecurity in IIoT environments is the timely detection and mitigation of cyber threats. Intrusion Detection Systems (IDS) play a pivotal role in safeguarding industrial systems against unauthorized access, malicious activities, and potential disruptions (Hamouda et al., 2021). The conventional approach of centralizing data analysis for IDS may, however, introduce latency and congestion, underscoring the need for innovative solutions such as distributed Machine Learning (ML) at the edge. In response to these challenges, edge computing has emerged as a transformative paradigm, aiming

[a] https://orcid.org/0000-0002-9691-9956
[b] https://orcid.org/0000-0003-4873-2705
[c] https://orcid.org/0000-0002-8239-5020
[d] https://orcid.org/0000-0002-2151-7944

Funchal, G., Pedrosa, T., de la Prieta, F. and Leitão, P.
Distributed Machine Learning and Multi-Agent Systems for Enhanced Attack Detection and Resilience in IoT Networks.
DOI: 10.5220/0013154400003899
In *Proceedings of the 11th International Conference on Information Systems Security and Privacy (ICISSP 2025) - Volume 2*, pages 192-203
ISBN: 978-989-758-735-1; ISSN: 2184-4356

to process data in proximity to its source, thereby reducing latency and mitigating network congestion.

According to the IBM report (IBM Security, 2023), cloud environments were frequent targets for cyber attackers in 2023, with 82% of breaches involving data stored in the cloud - public, private or multiple environments with 39% of breaches spanning multiple environments and incurring a higher than average cost of US$4.75 million. The report points out that only one-third (33%) of breaches/attacks were identified by the organizations' internal security teams and tools. Also, organizations with extensive use of security Artificial Intelligence (AI) and automation identified and contained a data breach 108 days faster than organizations with no use. This information reinforces the importance of keeping data closer to the source and using AI methods to improve the speed, accuracy and efficiency of attack detection. However, while edge computing addresses several challenges associated with data processing, it introduces new complexities, particularly in the context of ML tasks, where computational limitations can hinder the timely and effective detection of cyber threats in IIoT environments.

Having this in mind, this paper aims to search into strategies for effectively distributing ML tasks across multiple nodes with low computational power located at the edge, using Multi-agent systems (MAS) as infrastructure platform, focusing on their application in enhancing intrusion detection capabilities within Industry 4.0 and IIoT ecosystems. By exploring and comparing the different strategies, the research aims to provide insights into best practices for mitigating security risks associated with the exponential growth of connected devices and the evolving threat landscape in Industry 4.0. For this purpose, this work explores the following research questions:

RQ1: In which scenarios and under what conditions do distributed ML approaches outperform centralized ones in terms of attack detection accuracy, scalability, and resilience against adversarial attacks and novel threats?

RQ2: What are the trade-offs between model interpretability and detection accuracy in distributed training strategies for attack detection, and how can these trade-offs be optimized?

The findings of this study hold implications for advancing resilient and efficient ML applications in the pursuit of safeguarding critical industrial systems from cyber threats.

The remaining paper is organized as follows: Section 2 presents the related work, highlighting the importance of ML distribution at the edge, emphasizing the work developed in this domain, and especially outlining the gaps and challenges in this field. Section 3 presents the proposed approach to distributing ML at the edge by using MAS, summarizing its strengths and weaknesses. Section 4 describes the implementation and application of the proposed approach in a case study and discusses the achieved results. Finally, Section 5 rounds up the paper with the conclusions and points out some future work.

2 RELATED WORK

The generalized and exponential adoption of smart IoT devices, especially in industrial environments, is accelerating the search for new techniques to make IoT applications secure, scalable and energy-efficient (Alsboui et al., 2021). IoT devices typically come equipped with various sensors that gather environmental/operational data, serving as key components of data-driven intelligence systems (Kong et al., 2022). As these devices' deployment expands, the generated data volume grows exponentially. In order to provide insights to end users, it is necessary to process this collected data and analyze it first. Moreover, internet traffic between devices in an IoT network must be monitored, commonly by Network Intrusion Detection Systems (NIDS), acting as a first line of defense in order to identify potential threats and protect the network from malicious attacks and intruders (Gyamfi and Jurcut, 2022). However, most IoT devices have limited computing resources, making this processing a major challenge.

A commonly adopted solution is cloud computing, where IoT data is sent to remote servers for processing, and the results are transmitted back to the devices. While effective, this approach can face significant challenges in terms of data transmission rates and network bandwidth, which can become critical bottlenecks as IoT ecosystems scale (Shi et al., 2016). In addition, as IoT devices often handle personal and sensitive data, routing all information to cloud servers raises security and privacy concerns (Kong et al., 2022) and when these devices require a high service response time, it becomes a major challenge for cloud-based IoT applications (Quy et al., 2023).

In this context, edge computing is gaining attention, being a type of IT architecture in which data is processed at the edge of the network, or as close as possible to the data source, reducing costs and response times, increasing data privacy and security, and making it possible to make decisions in these network applications faster and with lower response latency (Quy et al., 2023). Despite the numerous advantages of edge computing over cloud computing,

it cannot fully replace cloud services (Ghosh et al., 2020; Wang et al., 2020a). While shifting analytics to the edge network is designed to reduce service response times, certain services still depend on cloud infrastructure. Moreover, the edge computing layer encounters several challenges, including task offloading, performance optimization, energy efficiency, Quality of Service (QoS) support, and connection management (Xie et al., 2019; Qadir et al., 2020).

Some benefits of edge computing were highlighted by (Quy et al., 2023), such as cost savings, backhaul traffic reduction, improved QoS, enhanced network customization, and improved service response times and distribution capabilities, which justify the growth in the adoption of edge computing. However, as the complexity of IoT applications increases, especially in environments such as IIoT, traditional centralized ML approaches become insufficient due to computing power constraints and the need for real-time analysis. This has led to the development and integration of distributed ML, where computational tasks are spread across multiple nodes, enabling large-scale data processing and model training closer to the data source.

Distributed ML offers the potential to overcome the computational bottlenecks that arise from handling large data sets and complex models at the edge. By distributing the training and inference processes across several edge nodes, it becomes possible to leverage the collective resources of the network, resulting in enhanced scalability, reduced latency, and improved adaptability to dynamic environments. Additionally, distributed ML techniques are particularly effective in scenarios where data privacy is crucial, as the data can remain localized at the edge nodes, minimizing the need for transmission to centralized servers. There are some strategies that can be used for distribution, such as data parallelism, model parallelism, ensemble learning/model combination, and model diversity. However, there are several issues that need to be analysed and addressed, namely minimal synchronization, communication overhead, device heterogeneity, security and privacy, among others (Duan et al., 2024; Khouas et al., 2024).

As highlighted in (Mwase et al., 2022; Wang et al., 2020b), data parallelism divides a batch of data into several smaller batches, and these input data samples are distributed among several computing resources (nodes or devices). It can thus improve the performance of large batch workloads. Each device performs local processing with its own data and the complete model, a copy of which is stored locally. Because the full model is present on every device, this structure works well with a wide range of model topologies and scales effectively when the model has few parameters. However, the parameter synchronization becomes a bottleneck when the model has many parameters (Jia et al., 2018).

On the other hand, (Mwase et al., 2022) notes that in the model parallelism a part of the model that each device uses for local processing is stored on it. Every device has a duplicate copy of the data. Due to its limited memory footprint and consequently low memory requirements on each device, this strategy works well when the model is too big to fit on a single device. Effective model splitting can be difficult, though, as a poorly done split can cause synchronization delays and communication overload, which can cause downtime (Mwase et al., 2022; Mirhoseini et al., 2017).

In addition to these strategies, there is ensemble learning (Zhou and Zhou, 2021), in which each node trains an independent model and then these models are combined to form a final model. This final model can be made up of different characteristics, which can enhance the results obtained by taking advantage of the strengths and reducing the weaknesses of each model. There are different possible techniques, such as bagging (or bootstrap aggregating), boosting and stacking that can be explored.

ML algorithms are used in IDS because of their capacity to recognize patterns, correlate events and adjust to emerging threats (Prazeres et al., 2023; Berman et al., 2019). The main benefits of using ML in IDS is the high anomaly detection accuracy, scalability to manage big data sets, and the potential to get better with time and new information. These technologies improve security by reducing false positives and providing real-time threat detection.

In this respect, many works have been developed in this area, such as the work proposed by (Mohyeddine et al., 2022), which highlights the use of ML algorithms, namely Random Forest, in which they propose an edge approach with methods for feature selection, dimensionality reduction and outlier removal in order to reduce computational costs and time, and improve performance in Industrial IoT intrusion detection, and also the work carried out by (Yang et al., 2023), that proposed a Temporal Convolutional Network (TCN) based intrusion detection method for IoT environments that is lightweight, effective and relies on cloud edge collaboration, based on a federated learning framework. To this end, they have also reduced the dimensionality of the high-dimensional resources of raw network traffic data to reduce computing and storage requirements while overcoming the problem of resource limitations of edge devices. (Yang et al., 2023) also carried out some experiments to validate that the collaboration-

based approach at the cloud edge can share threat intelligence and has the potential to defend against unknown attacks in a collaborative way, showing that collaboration was extremely important and could help participants identify their own unknown attacks.

However, there are still other distributed approaches, such as those based on MAS (Wooldridge, 2009), which act as containers/repositories for AI algorithms (Queiroz et al., 2019). According to (Leitão, 2009), an agent can be defined as an autonomous entity, which represents a physical or logical part of the system, and which will be able to take actions to achieve the system's objectives and will also be able to interact with other agents in the system when it does not have the skills to achieve its objectives alone. Thus, the use of MAS allows to distribute intelligence and adaptation (Leitão et al., 2016), in which decisions are made in a decentralized manner, as opposed to centralized structures that are unable to meet the requirements related to response time, data privacy and security, network bandwidth, among others. Although there is a certain complexity to coordinating the actions of various agents in a distributed environment, it becomes very advantageous to use it in dynamic systems, especially in IoT environments, where conditions change rapidly, there is a great heterogeneity of devices, which can use different algorithms due to their available computing resources.

Despite the fact that the aforementioned works have presented excellent solutions and results, different ways of distributing intelligence at the edge have not yet been explored, especially with regard to distribution using MAS, which can already be used for collaborative work in the edge network, and could also make the system more secure with an IDS based on ML. In this way, the proposed work aims to analyze the different strategies for distributing intrusion analysis and detection at the edge, using MAS as the basis for distribution. The main focus is on exploring the different strategies analyzed in a case study, highlighting the strengths and weaknesses of each strategy and summarizing the main conclusions reached.

3 DISTRIBUTED ATTACK DETECTION IN IoT

This section aims to describe the system architecture, the distribution strategies for ML algorithms for attack detection, and the MAS-based IDS.

3.1 System Architecture

The proposed system architecture is composed of two layers, edge and cloud, where detection nodes are distributed utilizing MAS technology, as shown in Figure 1. This architecture integrates multiple agents across both layers, with edge agents being responsible for monitoring specific processes or systems to ensure the correct functionality and detect anomalies caused by external attacks. In cases where edge agents are unable to independently execute detection or diagnosis tasks, they are capable of initiating cooperative interactions with other agents.

On the other hand, cloud agents manage the entire system, assisting edge agents in achieving their local goals. Due to their superior computational capabilities, they can execute more advanced ML algorithms, enabling the identification of complex system behavior patterns. Additionally, cloud agents can provide refined inferences regarding the parameters used in the models deployed in edge agents. Consequently, cloud agents can recommend updates to the ML models at the edge, enhancing the detection accuracy and the overall system performance. More details of this architecture and the interaction patterns between the agents can be found in (Funchal et al., 2024).

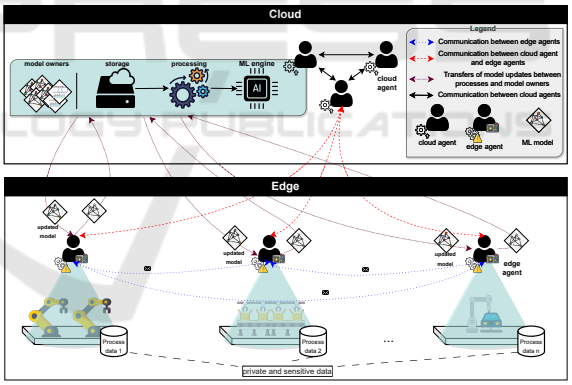

Figure 1: System architecture composed of agents distributed along the edge to cloud continuum (adapted from (Funchal et al., 2024)).

Although the architecture covers the edge to cloud continuum, the focus of this work is to analyze different strategies to use ML algorithms on edge devices, including how they will be trained, embedded and deployed together on edge devices.

3.2 ML Distribution Strategies

Distributing ML algorithms to obtain a distributed IDS can require a lot of analysis and searching for the best application that fits within the system's require-

ments. As a result, different strategies can be used to achieve the best performance in distributed detection. This means analyzing how to use the data for training, how to train the models, how to use different models on each device and how to aggregate the different results to achieve a final result in the case of collaboration between several detection nodes.

Among the different possibilities, when it comes to using the available data set to train the models, the data set can be divided into several parts, keeping the same proportions of each type of attack, in order to ensure that each training data set is representative of the overall distribution of the data, avoiding class imbalance that could harm the model's performance. These different data sets are used to train multiple models, which can help reduce training time and explore different subsets of the training data.

Another approach that can be adopted is to separate the data by different categories of attacks and train separate models for each category. With this, different types of attacks can have/present distinct characteristics, and this allows the models to specialize in detecting specific patterns associated with each type of attack, potentially improving the overall performance of the intrusion detection system.

Thus, three strategies will therefore be considered for analysis in this work:

- **Centralized Strategy:** train several models with all available data, select the best model and have it as a centralized approach.
- **Full Knowledge Strategy:** Train several models with different portions of data (all available data has been partitioned into several subsets of data that contain the same portion of each type of data/attack), select the best model for each subset of data, embed each model in a device and use them in a distributed and collaborative way.
- **Specialized Knowledge Strategy:** Train several models with different portions of data (all available data has been partitioned into several data subsets containing different classes of attack/data), select the best model for each data subset, embed each model in a device and use them in a distributed and collaborative way.

These strategies are modeled using set theory and mathematical notations to illustrate their similarities and differences.

Centralized Strategy

Let D be the entire data set, containing all types of attacks:

$$D = \{d_1, d_2, \ldots, d_N\}$$

In the centralized strategy, each model M_i is trained on the same complete data set D:

$$\hat{y}_i = M_i(D), \quad \text{for } i = 1, 2, \ldots, n$$

The model with the best performance, based on a pre-defined metric, is selected as the final model M^*. The final prediction $\hat{y}$ is then:

$$\hat{y} = M^*(X_{\text{new}})$$

where X_{new} represents new input data., and M^* is the model that performed best on the full data set D.

Full Knowledge Strategy

In the Full Knowledge Strategy, the data set D is divided into subsets $D_{i_1}, D_{i_2}, \ldots, D_{i_k}$:

$$D = D_{i_1} \cup D_{i_2} \cup \cdots \cup D_{i_k},$$

where each subset preserves the proportion of all types of attacks. Each model M_i is trained on a specific subset D_{i_j} of the data, which still contains knowledge about all types of attacks:

$$\hat{y}_i = M_i(D_{i_j}), \quad \text{for } i = 1, 2, \ldots, n$$

When a model M_i has low confidence in its prediction, it can request help from other models. The final prediction $\hat{y}$ is made based on the weighted predictions of all models:

$$\hat{y} = \frac{\sum_{i=1}^{n} w_i \hat{y}_i}{\sum_{i=1}^{n} w_i}$$

where w_i are the weights corresponding to the confidence levels C_i of each model's prediction.

Specialized Knowledge Strategy

In the Specialized Knowledge Strategy, the data set D is divided into disjoint subsets D_k based on different types of attacks or characteristics:

$$D = \bigcup_{k=1}^{m} D_k \quad \text{where } D_i \cap D_j = \emptyset \text{ for } i \neq j$$

where each subset contains only one category of attack. Each model M_k is trained only on the subset D_k, which contains data from a single type of attack:

$$\hat{y}_k = M_k(D_k), \quad \text{for } k = 1, 2, \ldots, m$$

If a model M_k requests help due to low confidence, the final decision $\hat{y}$ is derived from the collaborative predictions of all specialized models:

$$\hat{y} = \frac{\sum_{k=1}^{m} w_k \hat{y}_k}{\sum_{k=1}^{m} w_k}$$

where w_k are the weights corresponding to the confidence levels C_k of each specialized model.

Combining the results of several models can also be challenging. This combination can be done in different ways, either by voting, in which the majority of the decisions reached will be considered, or by a simple average of the decisions obtained. A weighted average can also be used, in which the weights are based on metrics such as the confidence of the predictions (e.g., accuracy of that decision), or by selecting the best decision among those obtained.

In order to improve the final decision made after receiving all predictions from all entities, the decision proposed by (Funchal et al., 2024) will be considered, in which an indicator function has been added so that only predictions with the stipulated minimum confidence weight are considered. In this way, the final decision will be given by the following Equation:

$$\hat{y} = \begin{cases} 1, & \text{if } \left(\frac{\sum_{i=1}^{n} (w_i \cdot \hat{y}_i \cdot \mathbb{I}(C_i \geq \beta))}{\sum_{i=1}^{n} w_i} \right) \geq \alpha \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In this extended decision-making process, an indicator function $\mathbb{I}$ is introduced, which ensures that only predictions with a confidence level greater than or equal to a predefined threshold β are considered. Specifically, for each model M_i, the function $\mathbb{I}(C_i \geq \beta)$ returns 1 if the confidence C_i meets or exceeds β, and 0 otherwise. The final decision $\hat{y}$ is then calculated based on the weighted sum of predictions that satisfy this confidence criterion. The decision threshold α determines whether the aggregated prediction leads to a positive classification (1) or a negative classification (0), as shown in Equation 1.

Thus, this method enhances the combination of results in distributed strategies by ensuring that only reliable predictions contribute to the final decision. This is particularly beneficial in distributed environments, such as those modeled with MAS, where the collaboration between different models or devices is essential for achieving robust predictions.

3.3 MAS-Based IDS Structure

The implementation of a distributed IDS on the edge requires that distributed devices have communication and collaboration capabilities, and present a scalable, efficient and secure structure to meet the demands of the Industry 4.0. To this end, the use of MAS fits these requirements perfectly, where each autonomous agent has the characteristics of communication and collaborative decision-making.

Detection nodes are then distributed along the edge cloud continuum by using MAS, in which each device has an embedded agent and ML algorithms. Figure 2 illustrates an overview of the agents distributed on the edge, which is the focus of this work. In general, it has several agents that may be in the same or different organizational relationships, but which are constantly acquiring knowledge of a given environment (data collection, whether it's from the process, a task, or the network to which the IoT device is connected), and depending on the complexity of the tasks or process, the agents can communicate with other agents that are linked to an organizational relationship or even with other agents linked to other processes in order to exchange information to meet the demands of the process. Interaction between agents depends on the needs of each one and the complexity of the task to be solved, so the agent can have a certain level of autonomy in consulting/requesting the collaboration of certain agents present in the system. In addition, the agents have the ability to run ML algorithms to recognize patterns in the data being monitored in that environment.

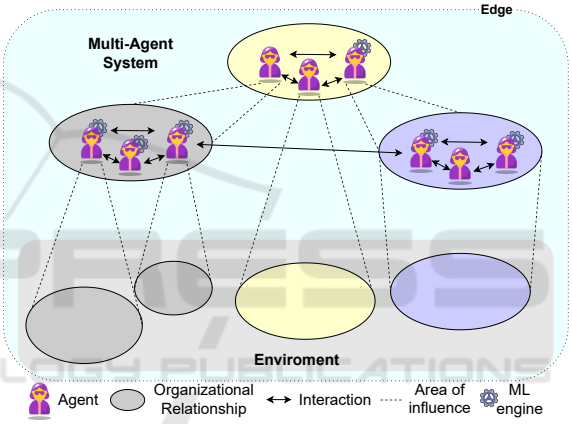

Figure 2: MAS structure (adapted from (Jennings, 2001)).

In practical terms, by having several detection nodes in the system, they can interact with each other to make better decisions when their self-analysis indicates that their decision may not be the best for that situation. Thus, this agent autonomy can facilitate adaptive responses to emerging threats and make the system more resilient to new threats. Figure 3 shows how the collaborative IDS is formed at the device edge, where the embedded systems are located.

Interaction between agents consists of exchanging messages for a specific purpose, which is structured according to the FIPA-ACL. The interaction pattern between the agents for intrusion detection is shown in Figure 4. The agents are autonomous in requesting collaboration, self-analyzing their results and, in case of uncertainty, they will request the collaboration of others to achieve a better prediction result.

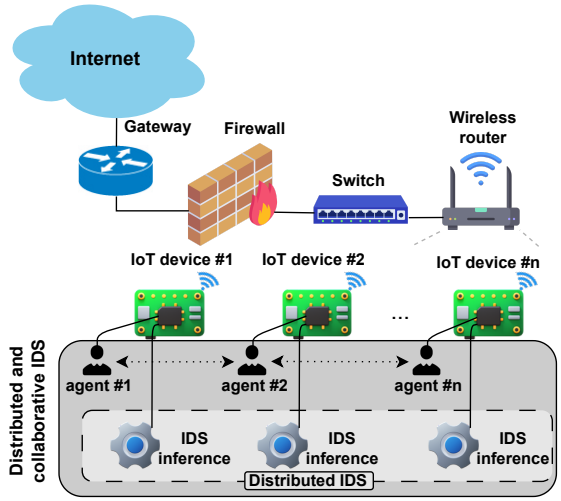

Figure 3: Distributed and collaborative IDS at the device edge.

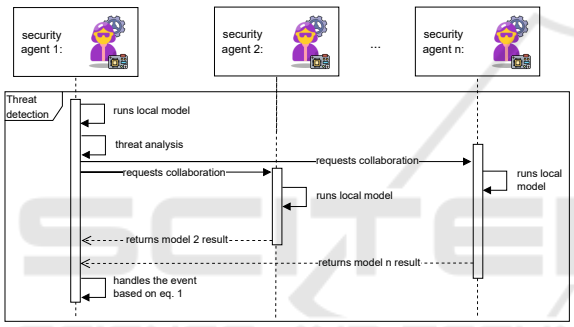

Figure 4: Interaction pattern for intrusion detection (based on (Funchal et al., 2024)).

4 EXPERIMENTAL IMPLEMENTATION

This section describes the implementation carried out to test the different strategies presented in the previous section, and the achieved results.

4.1 Data Preparation

As mentioned in the previous section, different strategies have been proposed to analyze the impact of collaboration on attack detection in an IoT environment. These strategies consist, in practical terms, of training different ML models with different subset of data, in order to select the best pairs (model-subset of data) to employ in each IoT node, so as to have a network of nodes with detection systems embedded at the edge, spread across a network, with communication characteristics/skills that will make it possible to have a collaborative and resilient IDS at the edge. For this purpose, it was necessary to structure the available data set, separating it in a viable way, and select lightweight models so that they are fast and efficient to run on devices with low computing power in IoT nodes at the edge.

The CICIoT2023 data set (Neto et al., 2023) was used, which contains data from 105 devices and 33 different types of attacks in IoT environments. With regard to the different data groups, to be used in the Specialized Knowledge Strategy, they have been separated (according to similarity) into five different categories, the first of which contains data related to Distributed Denial of Service (DDoS), Denial of Service (DoS) and Mirai attacks, the second contains Reconnaissance data, the third Spoofing, the fourth Brute force and the fifth web-based, as illustrated in Figure 5, which highlights the types of attack present in each category. It is important to note that in the case of the strategy that consists of using the same knowledge for all models (Full Knowledge Strategy), all categories were considered. Furthermore, in addition to the data that has been described, there is the set of data that is normal and benign to the system.

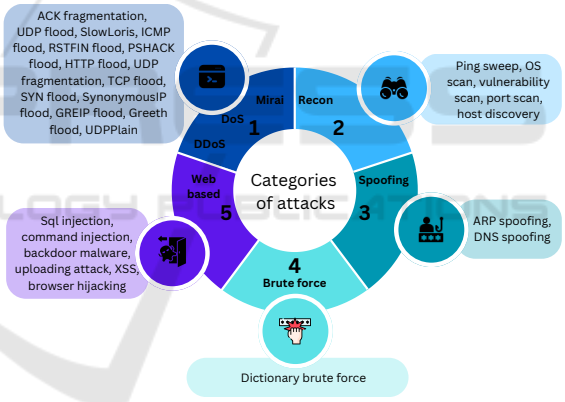

Figure 5: Categories of the different data sets for specialized training.

4.2 System Implementation

To implement the system, some ML algorithms were selected taking into account the processing restrictions of the computer platforms used, so that they do not cause bottlenecks during execution and are capable of returning good results. The models considered were Support Vector Machine (SVM), Random Forest (RF), Logistic Regression (LR), Gaussian Naive Bayes (GNB), and Convulational Neural Network (CNN), which were implemented using the scikit-learn library.

The data was split into 80% for training and 20% for testing. The part of the data for training was then

differentiated for the two distributed strategies mentioned, where for the Full Knowledge Strategy the Stratified Split was used to have several portions of data with the same amount of data representative of each type of attack, and for the Specialized Knowledge Strategy, the training data was separated according to the category of attacks (as shown in Figure 5). The test data was the same for all strategies.

The models were then trained with the data and the best models were selected (according to the observed accuracy, precision, recall and f1-score). For the Full Knowledge Strategy, the best performing models were selected for each data subset, for Specialized Knowledge Strategy the best performing models were selected for each attack category, and all models were saved using the Joblib tool. The saved models were then embedded in the devices so that inferences could be made directly on the devices at the edge, namely in the Raspberry Pi 3 model b+, to implement the distributed and collaborative IDS.

The core of the proposed distribution is based on MAS. For this purpose, the MAS was implemented by using the JADE framework, which individual behavior and interact patterns are described in (Funchal et al., 2024) and in Figure 4. The agents are then embedded in IoT devices (Raspberry Pi 3), together with the ML models in order to have a distributed and collaborative IDS system, as illustrated in Figure 6.

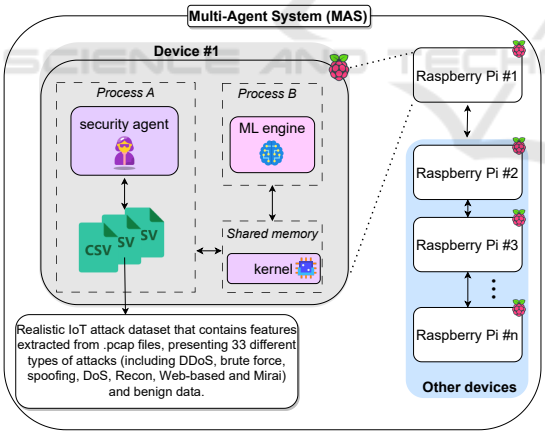

Figure 6: Structure of edge devices with collaboration and attack detection capabilities.

Each device contains two processes, the security agent and the ML engine. The function of the security agent is to acquire network data, which in this case is done by reading the data that has already been collected (CICIoT2023 data set), extracting the main features and sending this data to the second process (ML engine) to predict whether the data is normal or an attack. For this communication, the two processes exchange information via a shared memory in the kernel, through the Unix socket.

The security agent also has the ability to communicate and collaborate with other agents, where messages are exchanged over the network via TCP/IP. Thus, on receiving the prediction made by the ML engine, it is evaluated by the agent, which will analyze the reliability of the prediction and take the action of requesting help when necessary. When it needs help due to the uncertainty of its prediction, it will ask the other agents to evaluate the selected data and will receive the predictions of the other agents in the system together with their confidences to make a final decision, which is based on Equation 1. Different ways of aggregating the different results were also compared, analyzing the results obtained through a simple vote, a simple average and the weighted average (Equation 1), the latter being considered for the analysis of the strategies because it showed better results than the others. In this way, the IDS becomes collaborative and distributed at the edge.

4.3 Evaluation Metrics

The metrics used to evaluate the performance of the ML algorithms were calculated, namely accuracy, recall, precision, f1-score and ROC-AUC. In addition, the number of collaboration requests in each strategy was analyzed, as well as the number of samples that had collaboration, and consequently the number of samples that had the prediction error avoided due to collaboration.

In short, accuracy parameter measures the proportion of correct predictions made by the model across the entire data set/samples, as shown in Equation 2.

$$\text{accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

where the True Positive (TP) indicates the correct identification of an attack, True Negative (TN) indicates the correct identification of normal data, False Positive (FP) indicates the incorrect identification of normal data (normal traffic) and False Negative (FN) indicates the incorrect identification of an attack.

Accuracy is a good metric when the different classes are well balanced. However, when there is unbalance between classes, it can give a false impression of good performance. When dealing with attacks, especially DoS/DDoS, accuracy can be used to measure the improvement of one technique over another, but care must be taken with the performance obtained and it is necessary to analyze other metrics together.

Precision, on the other hand, measures the proportion of true positive predictions among all positive predictions made by the model, focusing on the quality of positive predictions, as shown in Equation 3.

$$\text{precision} = \frac{TP}{TP+FP} \tag{3}$$

Recall, also known as sensitivity, measures the proportion of true positive predictions among all actual positive instances, focusing on the quality of negative predictions, as shown in Equation 4.

$$\text{recall} = \frac{TP}{TP+FN} \tag{4}$$

The evaluation of the f1-score metric consists of calculating the harmonic mean of precision and recall, and is used when a balance is sought between high precision and high recall. If one of the components (precision or recall) shows low values, this metric will be punished, clearly indicating the disparity between them. Equation 5 shows the calculation for f1-score.

$$\text{f1-score} = \frac{2 \times precision \times recall}{precision + recall} \tag{5}$$

Another metric widely used to evaluate model performance is the ROC-AUC score, which summarizes a model's capacity to generate relative scores to distinguish between positive or negative instances at all classification thresholds. The score goes from 0 to 1, where 0.5 indicates random guessing and 1 indicates perfect performance.

4.4 Discussion of Results

The achieved results are shown in Table 1, which contains data on the accuracy, recall, precision, f1-score, and ROC-AUC of the three strategies tested, namely collaboration with full knowledge, collaboration with specialized knowledge and the centralized approach where there is no collaboration. In addition, for the two approaches with collaboration, the number of samples that required collaboration and the number of errors avoided (wrong predictions that were corrected with collaboration) were also evaluated.

The results show that the rate of collaboration requests in the specialized knowledge strategy is much higher than in the full knowledge strategy, being four times higher, which was already expected due to the fact that each agent has specific knowledge and could therefore rely more on other agents to help them make decisions. But when analyzing the percentage of requests in both approaches, it shows that they had no more than 4.3% of requests, which is an acceptable amount in relation to not being in doubt about the decisions to be made and always asking for help, which could block the whole system with too many requests. Although the specialized knowledge strategy made more requests for collaboration, it had much more impact on correcting wrong predictions, with a rate of 32.1%, indicating that specialized knowledge collaboration prevented many errors in predictions. On the other hand, the full knowledge strategy had 7.7% in correcting wrong predictions, also contributing significantly. The large difference between these two values can be justified by the fact that having specialized knowledge is subject to more errors due to different characteristics that have not been learned by the model, and will therefore be corrected through collaboration. In a way, this makes this approach more resilient in detecting new attacks.

Another important aspect is the memory used by these models in these different approaches, which is a sensitive point when it comes to an edge approach. Specialized knowledge may be the best choice when there are many memory constraints, because it uses less data to train each model, optimizing the memory used by the model.

Regarding the performance metrics of each approach, it can be seen that the collaborative approaches performed better than the centralized one, although they did not show a higher accuracy value. This can be explained by the fact that accuracy is a global metric that measures the proportion of all correct predictions. However, it can be insensitive to unbalanced classes (there is much more attack data than normal data, mainly due to DoS and DDoS attacks) or to specific cases where precision and recall are more critical (scenarios with attacks). Thus, the improvement in all other metrics, especially f1-score and ROC-AUC, indicates that collaborative approaches are doing a better job of correctly identifying the instances of interest, even if this does not directly increase overall accuracy.

In relation to the specialized knowledge strategy, the models can be better adjusted to capture nuances in the data, which can directly result in improvements in precision and recall metrics due to better detection of specific patterns. It should be borne in mind that the increase in precision and recall reflects directly on the correction/reduction of false positives and false negatives, which can have a significant cost for an IDS, where attacks are considered as normal data and are not identified, and normal data are identified as attacks and start to interfere in the analysis when having to mitigate the attacks (e.g. in the next step with the implementation of an Intrusion Prevention System (IPS)). Figure 7 summarizes the improvement of the different approaches due to collaboration. It can be seen that the collaborative approaches contributed significantly to improving precision, recall, f1-score and ROC-AUC compared to the centralized one, showing improvements between 2.3% and 3.3%.

Table 1: Detailed comparison of distributed strategies and centralized approach.

Attribute	Full Knowledge Strategy	Specialized Knowledge Strategy	Centralized Strategy
Total Samples Analyzed	9507466		
Samples Requiring Collab.	98043 (1.03%)	407269 (4.28%)	-
Errors Avoided by Collab.	7574 (7.73%)	131069 (32.18%)	-
Memory Usage - model size	13748 KB	5799 KB	15900 KB
Accuracy	0.9952	0.9948	0.9968
Recall	0.9970	0.9950	0.9645
Precision	0.9981	0.9997	0.9679
F1-Score	0.9976	0.9974	0.9662
ROC-AUC	0.9904	0.9918	0.9677

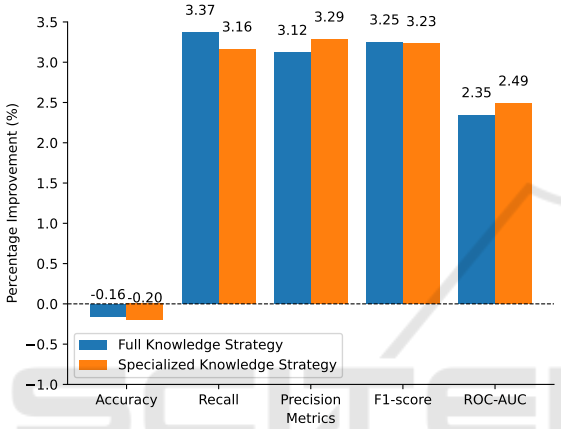

Figure 7: Percentage improvement for the different distributed strategies compared to the centralized one.

4.5 Key Findings

Considering the analysis of the results obtained in the case study and with the study of the literature, it was possible to summarize the main key findings in distributing the ML algorithms, listed in Table 2, where some important aspects were considered when having more robust algorithms (strong algorithm) and algorithms with more restrictions (weak algorithm), and the impact observed. In terms of memory usage, the better the algorithm chosen (within the limitations and restrictions found), the less CPU memory will be used, as fewer collaborations will be needed, reducing the amount of message/knowledge exchange.

The use of RAM, on the other hand, will depend directly on the platform on which it is being used. Linked to this, the weaker the algorithm used due to the restrictions, the greater the requests for collaboration, which will consequently increase the communication load. In terms of improved detection, algorithms that are not very powerful show a significant improvement, while algorithms that already perform very well require little collaboration, but still show improvement because with collaboration they are able to correct the small errors they would have.

More powerful algorithms have a lower probability of error and, on the contrary, weak algorithms have a high probability of error if they don't collaborate. With the collaboration mechanism, this probability is significantly reduced because they start to use the strengths of the other algorithms. In terms of scalability, if the collaboration network has many weak algorithms, it can generate many bottlenecks, while if they are strong, requests will be made to correct small errors in each of the algorithms, making it highly scalable for large-scale systems.

5 CONCLUSIONS

This paper addresses various challenges within the IoT ecosystem when processing data remotely from its source. These challenges include response time, network bandwidth, data transmission rate, and security concerns. Security is particularly critical for IoT devices, especially in terms of threat detection within this environment. Consequently, there is a need to bring data processing as close to the data source as possible to achieve faster and more efficient responses. To address these requirements, this study proposes a distributed and collaborative IDS at the edge, specifically on IoT devices. This system utilizes MAS for distribution and collaboration, along with ML algorithms to identify data patterns collected from the network. To assess the most effective methods for distributing the IDS at the edge, various strategies were studied and evaluated using performance metrics for ML algorithms. This analysis aimed to highlight the strengths and weaknesses of each approach. The well-known CICIoT2023 data set was used to validate the proposed approach, which provides real data from a large environment of diverse IoT devices and a variety of attacks on a network.

Table 2: Key findings in the distribution and collaboration of ML algorithms.

Aspect	Strong Algorithm	Weak Algorithm	Impact
Memory Usage	Low	High	Weak algorithms will use more processor memory because they will require more collaboration. On the other hand, specialized agents use less memory to store models, while generalist agents need more.
Collaboration Requests	Low	High	Weak algorithms request more collaboration, leading to communication overhead.
Accuracy (Without Collab.)	High	Low	Strong algorithms have good standalone accuracy.
Accuracy (With Collab.)	Slight improvement	Significant improvement	Weak algorithms improve more with collaboration but may overly rely on it.
Effective Final Decision	Weighted Average	Simple Voting	Weighted decisions are more robust when collaboration is frequent.
Communication Overhead	Low	High	More collaboration leads to higher data traffic.
Processing Time	Standard	High	Evaluate if edge network overhead is significant.
Error Probability	Low	High	Weak algorithms may misjudge, leading to higher error rates without sufficient collaboration.
Edge Network Impact	Low	High	Network delays may occur depending on the volume of communication between agents.
Scalability	High	Limited	Strong algorithms are better suited for handling larger-scale systems with multiple agents.
Energy Consumption	Low	High	Collaboration requires more energy in IIoT environments.
Platform Constraints	Low	High	Stronger algorithms are easier to embed, while weak algorithms may have limitations on specific hardware.

The proposed approach, which includes different distribution and collaboration strategies, has shown promising results, indicating that distribution and collaboration outperform centralized structures, are more resilient to new threats, and significantly reduce the error rate in predictions. In general terms, the strategies with collaborative IDS showed between 2.35% and 3.37% improvement in precision, recall, F1-score and ROC-AUC. Also, among the predictions that had low confidence, a correction of these predictions of between 7.73% and 32.18% was observed, indicating the great potential of the approaches analyzed. Overall, it was possible to summarize the strengths and weaknesses of each approach, bringing out the important impacts of each choice and system constraints.

Future work will be dedicated to extend the analysis of the ML distribution for a distributed IDS at the edge, in order to further optimize the models to be employed on computational constrained devices. In addition, collaboration between cloud and edge agents will be studied taking advantage of MAS to build a more efficient and resilient IDS.

ACKNOWLEDGEMENTS

This work has been supported by national funds through FCT/MCTES (PIDDAC): CeDRI, UIDB/05757/2020 (DOI: 10.54499/UIDB/05757/2020) and UIDP/05757/2020 (DOI: 10.54499/UIDP/05757/2020); and SusTEC, LA/P/0007/2020 (DOI: 10.54499/LA/P/0007/2020). The author Gustavo Funchal thanks the FCT Portugal for the PhD Grant 2022.13712.BD.

REFERENCES

Alsboui, T., Qin, Y., Hill, R., and Al-Aqrabi, H. (2021). Distributed Intelligence in the Internet of Things: Challenges and Opportunities. *SN Computer Science*, 2(4).

Berman, D. S., Buczak, A. L., Chavis, J. S., and Corbett, C. L. (2019). A Survey of Deep Learning Methods for Cyber Security. *Information*, 10(4).

Duan, J., Zhang, S., Wang, Z., Jiang, L., Qu, W., Hu, Q., Wang, G., Weng, Q., Yan, H., Zhang, X., Qiu, X., Lin, D., Wen, Y., Jin, X., Zhang, T., and Sun, P. (2024).

Efficient Training of Large Language Models on Distributed Infrastructures: A Survey.

Funchal, G., Pedrosa, T., Prieta, F. D. L., and Leitao, P. (2024). Edge Multi-agent Intrusion Detection System Architecture for IoT Devices with Cloud Continuum. In *2024 IEEE 7th ICPS*.

Ghosh, S., Mukherjee, A., Ghosh, S. K., and Buyya, R. (2020). Mobi-IoST: Mobility-Aware Cloud-Fog-Edge-IoT Collaborative Framework for Time-Critical Applications. *IEEE Transactions on Network Science and Engineering*, 7(4):2271–2285.

Gyamfi, E. and Jurcut, A. (2022). Intrusion Detection in Internet of Things Systems: A Review on Design Approaches Leveraging Multi-Access Edge Computing, Machine Learning, and Datasets. *Sensors*, 22(10):3744.

Hamouda, D., Ferrag, M. A., Benhamida, N., and Seridi, H. (2021). Intrusion Detection Systems for Industrial Internet of Things: A Survey. In *2021 ICTAACS*.

IBM Security (2023). Cost of a Data Breach Report. https://www.ibm.com/reports/data-breach.

Jennings, N. R. (2001). An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41.

Jia, Z., Zaharia, M., and Aiken, A. (2018). Beyond Data and Model Parallelism for Deep Neural Networks.

Kagermann, H., Wahlster, W., and Helbig, J. (2013). Securing the Future of German Manufacturing Industry: Recommendations for Implementing the Strategic Initiative INDUSTRIE 4.0. Technical report, ACATECH.

Khouas, A. R., Bouadjenek, M. R., Hacid, H., and Aryal, S. (2024). Training Machine Learning models at the Edge: A Survey.

Kong, L., Tan, J., Huang, J., Chen, G., Wang, S., Jin, X., Zeng, P., Khan, M., and Das, S. K. (2022). Edge-computing-driven Internet of Things: A Survey. *ACM Computing Surveys*, 55(8):1–41.

Leitão, P. (2009). Agent-based distributed manufacturing control: A state-of-the-art survey. *Engineering Applications of Artificial Intelligence*, 22(7):979–991. Distributed Control of Production Systems.

Leitão, P., Colombo, A. W., and Karnouskos, S. (2016). Industrial automation based on cyber-physical systems technologies: Prototype implementations and challenges. *Computers in Industry*, 81:11–25. Emerging ICT concepts for smart, safe and sustainable industrial systems.

Mirhoseini, A., Pham, H., Le, Q. V., Steiner, B., Larsen, R., Zhou, Y., Kumar, N., Norouzi, M., Bengio, S., and Dean, J. (2017). Device Placement Optimization with Reinforcement Learning.

Mohy-eddine, M., Guezzaz, A., Benkirane, S., and Azrour, M. (2022). An effective intrusion detection approach based on ensemble learning for IIoT edge computing. *Journal of Computer Virology and Hacking Techniques*, 19(4):469–481.

Mwase, C., Jin, Y., Westerlund, T., Tenhunen, H., and Zou, Z. (2022). Communication-efficient distributed AI strategies for the IoT edge. *Future Generation Computer Systems*, 131:292–308.

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., and Ghorbani, A. A. (2023). CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment. *Sensors*, 23(13).

Popescu, D., Zilberman, N., and Moore, A. (2017). Characterizing the impact of network latency on cloud-based applications' performance. Technical report.

Pourrahmani, H., Yavarinasab, A., Monazzah, A. M. H., and Van herle, J. (2023). A review of the security vulnerabilities and countermeasures in the Internet of Things solutions: A bright future for the Blockchain. *Internet of Things*, 23:100888.

Prazeres, N., de C. Costa, R. L., Santos, L., and Rabadão, C. (2023). Engineering the application of machine learning in an IDS based on IoT traffic flow. *Intelligent Systems with Applications*, 17:200189.

Qadir, J., Sainz-De-Abajo, B., Khan, A., García-Zapirain, B., De La Torre-Díez, I., and Mahmood, H. (2020). Towards Mobile Edge Computing: Taxonomy, Challenges, Applications and Future Realms. *IEEE Access*, 8:189129–189162.

Queiroz, J., Leitão, P., Barbosa, J., and Oliveira, E. (2019). Distributing Intelligence among Cloud, Fog and Edge in Industrial Cyber-physical Systems. In *Proceedings of the 16th ICINCO*. SCITEPRESS - Science and Technology Publications.

Quy, N. M., Ngoc, L. A., Ban, N. T., Hau, N. V., and Quy, V. K. (2023). Edge computing for real-time Internet of Things applications: Future internet revolution. *Wireless Personal Communications*, 132(2):1423–1452.

Shi, W., Cao, J., Zhang, Q., Li, Y., and Xu, L. (2016). Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646.

Tyagi, S. and Tyagi, A. (2024). *Edge Computing: Empowering Real-Time Data Processing and Connectivity*.

Wang, H., Liu, T., Kim, B., Lin, C., Shiraishi, S., Xie, J. L., and Han, Z. (2020a). Architectural Design Alternatives based on Cloud/Edge/Fog Computing for Connected Vehicles. *CoRR*, abs/2009.12509.

Wang, Y. E., Wu, C.-J., Wang, X., Hazelwood, K., and Brooks, D. (2020b). Exploiting Parallelism Opportunities with Deep Learning Frameworks.

Wooldridge, M. (2009). *An Introduction to MultiAgent Systems*. Wiley.

Xie, R., Tang, Q., Wang, Q., Liu, X., Yu, F. R., and Huang, T. (2019). Collaborative Vehicular Edge Computing Networks: Architecture Design and Research Challenges. *IEEE Access*, 7:178942–178952.

Yang, R., He, H., Xu, Y., Xin, B., Wang, Y., Qu, Y., and Zhang, W. (2023). Efficient intrusion detection toward IoT networks using cloud–edge collaboration. *Computer Networks*, 228:109724.

Zhou, Z.-H. and Zhou, Z.-H. (2021). *Ensemble learning*. Springer.