IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 7 August 2023, accepted 31 August 2023, date of publication 4 September 2023, date of current version 11 September 2023.

*Digital Object Identifier 10.1109/ACCESS.2023.3311822*

RESEARCH ARTICLE

# A Collaborative DNN-Based Low-Latency IDPS for Mission-Critical Smart Factory Networks

**POULMANOGO ILLY[1] AND GEORGES KADDOUM[1,2], (Senior Member, IEEE)**
[1]Electrical Engineering Department, École de Technologie Supérieure, Montreal, Quebec H3C 1K3, Canada
[2]Cyber Security Systems and Applied AI Research Center, Lebanese American University, Beirut 1102, Lebanon

Corresponding author: Poulmanogo Illy (poulmanogo.illy.1@ens.etsmtl.ca)

This work was supported by NSERC under Grant B-CITI CRDPJ 501617-16.

**ABSTRACT** Industrial Control Systems (ICSs) have entered an era of modernization enabled by the recent progress in Information Technologies (IT), particularly the Industrial Internet of Things (IIoT). This enables better automation of industrial processes but now exposes the ICSs to cyber-attacks that exploit the IIoT vulnerabilities. Thus, to ensure ICSs security, numerous research works have focused on designing Intrusion Detection and Prevention Systems (IDPSs), and deep learning has recently received considerable attention, as it has the potential to improve detection accuracy. However, most of the proposed deep learning solutions focus only on the model's accuracy without considering latency, which is an essential requirement in many ICSs. The novelty of this paper is the time complexity analysis of Deep Neural Networks (DNNs) and the design of a low latency and robust deep learning-based collaborative IDPS. The proposed architecture employs two classification models. In the first model, a lightweight DNN is used to perform a binary classification, *i.e.*, normal or attack, which ensures rapid intrusion detection. A second model ensures the identification of the type of attacks by performing a multi-class classification of the detected anomaly, which is handled by a robust and complex DNN in order to achieve higher accuracy. This research also proposes intrusion response measures to deal with detected attacks, first after the anomaly detection, and then after the identification of the attack type. An experimental evaluation has been provided using various detection features, datasets, DNN algorithms, and the results demonstrate the effectiveness of the proposed solution.

**INDEX TERMS** Deep learning, industrial control system (ICS), industrial Internet of Things (IIoT), intrusion detection system (IDS), intrusion response system (IRS), network security, smart factory.

## I. INTRODUCTION

Industrial facilities are usually highly delicate and risky environments, requiring maximum safety and security to work with potentially dangerous chemicals and tools, and privacy to manufacture highly competitive products. Therefore, many security, safety, and privacy standards have been implemented over the years to protect these environments [1], [2], [3]. However, industrial accidents and disasters still occur frequently worldwide and cause significant damages, including deaths, injuries, economic losses, and long-term environmental impacts. According to the survey published in June 2021 by the research department of the database company Statista, there were 984 explosive incidents across the United States (USA) in 2020, which is a significant increase compared to previous years [4]. The sources of these explosions include manufacturing plants, electric utilities, petroleum industries (upstream, midstream, downstream, pipelines), and other chemical factories. The evolution of the industrial domain towards the new era of modernization driven by the Industrial Internet of Things (IIoT) enables real-time safety applications to prevent the traditional risk of incidents. However, it generates new security risks, notably from cyber-attacks that exploit the vulnerabilities of the connected objects.

Unlike traditional standalone Industrial Control Systems (ICSs), which used to be isolated from Information and Communication Technology (ICT) networks, new ICSs integrate these networks to enable high-level process supervisory management. Indeed, thanks to connected sensors, actuators,

The associate editor coordinating the review of this manuscript and approving it for publication was Varuna De Silva.

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

and IIoT devices, manufacturing equipment (pumps, valves, compressors, tanks, etc.) and manufacturing conditions (air quality, humidity, temperature, etc.) can now be easily monitored and controlled remotely using computers, tablets, or smartphones. For more efficient control systems operation (better resource allocation, easier and faster data collection), many industries are adopting the Supervisory Control And Data Acquisition (SCADA) architecture. This architecture is generally made up of three main levels. The first level consists of sensors and actuators, where the sensors collect data about the system, and the actuators control the system's state. The second level is composed of Programmable Logic Controllers (PLCs), which are connected to the first level in order to control the actuators and collect information from the sensors. The third level contains the supervisory controls, which communicate with the PLCs to send control commands from workstations, store the system data in servers (data historians), and provide a visual representation of the system on a Human-Machine Interface (HMI). Fig. 1 illustrates a generic SCADA system.

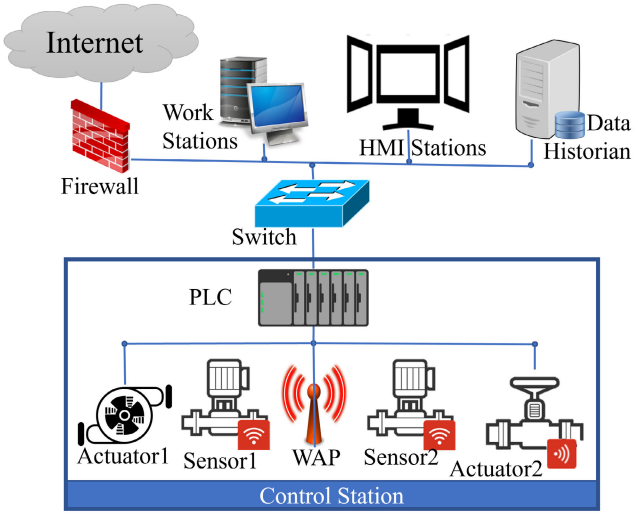


**FIGURE 1.** Generic SCADA architecture.

In terms of industrial safety and security, the adoption of ICT, especially IIoT, brings a lot of opportunities and challenges. For instance, a factory safety system integrated with the factory automation system can provide additional safety services, such as personalized alerts (*e.g.*, material physical default, place humidity, abnormal temperature, etc.), access control, occupancy detection, person identification, central locking of all perimeter doors and windows, remote surveillance of security cameras and sensors over the Internet, and others. However, the factory's IIoT can bring significant security problems to the factory because of the security vulnerabilities in IoT devices and software that hackers could leverage to carry out malicious activities. For example, attackers have found vulnerabilities in Siemens Step7, a software used in PLCs, and exploited it to launch an attack named Stuxnet. Stuxnet collected surveillance data, put ICSs into a critical state, and even falsely responded to prevent alarms [5]. A successful cyber-attack on a smart factory may have critical consequences, including private data leakage and cyber-terrorism.

### A. RELATED WORK

To overcome smart ICSs' cyber-security challenges, the research community is active in designing solutions for different security layers, especially Intrusion Detection Systems (IDSs), to detect malicious attempts (succeeded or failed) to penetrate IIoT networks. For example, Beaver et al. [6] applied machine learning algorithms to detect malicious Remote Terminal Unit (RTU) communications. The authors used a dataset composed of labelled RTU telemetry data from a gas pipeline system in the Mississippi State University's Critical Infrastructure Protection Center, USA. In this dataset, variants of command injection and data injection attacks were considered. The authors implemented six different machine learning algorithms, including Naive Bayes, Random Forests, One Rule (OneR), J48 (a type of Decision Tree technique), Non-Nested generalized exemplars (NNge), and Support Vector Machines (SVM). Their experimental results demonstrated the ability of the learning algorithms to detect these attacks. Lin et al. [7] established an industrial control system test-bed where they examined operational cases and developed a Modbus/Transmission Control Protocol (TCP) network attack program. They executed dozens of penetration attacks successfully, including address and function code scans, response and command injections, and Denial of Service (DoS). All network activity, including records of normal behavioral patterns, was collected. Exploiting content from the data link layer through the application layer, especially, Medium Access Control (MAC) and Internet Protocol (IP) addresses, TCP ports, and Modbus functions and data, the authors planned as future research to build on machine learning-based detection models, *e.g.*, a one-class SVM to find outliers, and thus effectively detect the occurrence of abnormal events. Teixeira et al. [8] developed a SCADA system test-bed where they conducted sophisticated cyber-attacks, including port and address scans, device identification attacks, and exploits. The authors captured the network traffic during the attacks, extracted features, and built a dataset for training and testing different machine learning algorithms. Five shallow machine learning algorithms were trained to detect the attacks, namely, Random Forest, Decision Tree, Logistic Regression, Naïve Bayes, and K-Nearest Neighbors (KNN). Their evaluation results showed the efficiency of the machine learning models in detecting the attacks in real-time.

To achieve low computational complexity and latency in anomaly-based intrusion detection models for SCADA networks, Ullah and Mahmoud [9] implemented a feature selection filter based on the information gain. Using an industrial control system dataset developed at the Distributed Analytics and Security Institute at Mississippi State University, the

proposed model selected a subset of five features out of the 20 initial features.

The dataset was used to train a J48 classifier. Then, then a Bayesian network classifier was used to develop the proposed model, which correctly classified all instances of the binary-labelled and categorized-labelled datasets. An Le et al. [10] proposed an Intrusion Detection and Prevention System (IDPS) that leveraged the Software-Defined Networking (SDN) approach to reduce the cost and decrease detection and mitigation latency. In the proposed system, the authors employed a C4.5 Decision Tree algorithm to build the detection model. The data used to train the model included 25 basic features (packet headers) and derived features (features that are computed). The system was evaluated using the probing and DoS attacks present in the 1999 DARPA dataset and a small test-bed where they generated attacks consisting of 3 types of DoS and 8 types of Probe. In the deployment, OpenFlow switches were implemented to replace traditional switches, IDPS sensors, and the firewall, to reduce the total cost of the IDPS.

Most of these solutions are based on shallow machine learning methods; therefore, they may suffer from significant limitations associated with shallow learning. To overcome these limitations, recent research works investigate deep learning-based IDSs. For example, Al-Abassi et al. [11] demonstrated an attack detection model that leveraged Deep Neural Networks (DNNs) and Decision Tree classifiers to detect cyber-attacks in an ICS environment. Li et al. [12] designed a deep learning-based intrusion detection model by making use of a Convolutional Neural Network (CNN) and a Gated Recurrent Unit (GRU). Then, the authors developed a federated learning framework, allowing multiple ICSs to collectively build a comprehensive intrusion detection model in a privacy-preserving way. Ling et al. [13] studied the limitations of intrusion detection methods based on deep learning, such as Long Short-Term Memory (LSTM) and GRU, to highlight problems these methods are still facing, such as vanishing gradients and low training efficiency. Then, the authors proposed an intrusion detection method based on a Bidirectional Simple Recurrent Unit (BiSRU). With skip connections employed, the optimized bidirectional structure in the SRU neural network alleviated the vanishing gradient problem and improved the training effectiveness. The author in [14] proposed an Attention-based Bi-Directional Gated Recurrent Neural Network (ABi-GRNN) model with a Poor and Rich Optimization algorithm-based hyper-parameter optimizer to build an efficient IDS for cyber-physical systems. The proposed system applied blockchain technology to boost security in the cyber-physical environment. The solution was evaluated using the NSL-KDD and CICIDS datasets, and the experimental results showed better precision compared to other DNNs, such as GRU and optimal GRU.

Although instructive, most of these deep learning-based IDSs for ICSs do not analyze the time complexity of DNNs and the latency of the proposed detection models. To enable a timely identification of various attacks and near real-time neutralization of threats, Shafi et al. [15] proposed a fog-assisted SDN and blockchain-driven IDPS for IoT networks. The authors employed three DNN-based classifiers, namely, Recurrent Neural Network (RNN), Multi-Layer Perceptron (MLP), and Alternate Decision Tree (ADT) in parallel with a voting system to identify attacks at the edge network just beside IoT devices. This approach reduces the detection latency by bringing the detection system as near as possible to the edge devices. However, unlike our approach, all classification models are deployed in cloudlet or fog nodes, and no advantage is taken from cloud resources. Alkadi et al. [16] proposed a distributed IDS that employed a Bidirectional Long Short-Term Memory (BiLSTM) deep learning algorithm to deal with sequential network data. The proposed system implemented a blockchain and smart contract method to provide privacy to the distributed intrusion detection engines. This approach provides security and concurrently ensures data privacy in cloud environments but does not address latency concerns for mission-critical IoT applications.

In addition to the lack of time complexity analysis for most DNN-based IDSs for ICSs, most previous studies do not consider attacks classification and intrusion response methods when attacks are detected, or the deployment architecture of the deep learning-based IDSs within the ICS networks.

### B. CONTRIBUTIONS

The novelty of this study is in its effort to design a DNN-based IDPS that preserves mission-critical ICSs requirements. Mission-critical ICSs refer to IIoT applications that require high availability and low latency to ensure real-time operations. In such applications, security measures must be designed to preserves the low latency requirement. Therefore, this study proposes an intrusion detection and response system that combines robustness and low latency. To achieve robustness, this work identifies various detection features and employs the most promising DNNs to build the detection models. To meet the ICSs' low latency requirement, the structure of the neural networks and their time complexity are studied. Then, a collaborative scheme that separates the classification task into two consecutive models organized based on a priority principle in the IDPS is implemented. The first model, based on a lightweight DNN, performs anomaly detection on local servers to allow timely attack detection and response. The second model performs attack classification of the anomalous traffic to help choose the suitable intrusion response measures to stop the attack. This multi-class classifier is deployed in cloud servers to benefit from the high computation resources available for running complex DNNs. The major contributions of the proposed work are four-fold:

- This paper proposes a time complexity analysis of the DNNs and highlights the variables that impact the training and the prediction latency.

- A collaborative deep learning-based IDS that employs two classification models is proposed along with its deployment architecture in the ICS.
- Various detection features, DNNs algorithms, and Intrusion Response System (IRS) measures are proposed to implement the IDPS.
- An experimental evaluation is conducted using different datasets, which demonstrates the efficiency of the proposed approach.

### C. ORGANIZATION

The rest of the research paper is organized as follows. Section II introduces the time complexity analysis of the DNNs and highlights the variables that impact the training and the prediction latency. Section III illustrates the collaborative deep learning-based IDPS and the deployment architecture. Section IV presents the detection features, learning algorithms, and IRS measures. Section V presents the experimental evaluation and discusses the obtained results. Finally, Section VI concludes this work.

## II. TIME COMPLEXITY ANALYSIS OF DEEP NEURAL NETWORKS

The time complexity in the training and prediction of the DNNs are proportional to the number and size of elementary structures and functions involved in their networks, *i.e.*, the neurons, recursions, convolutions, pooling, etc. This section evaluates how some of these elementary structures increase training and detection latency. In order to understand the latency reduction scheme, this evaluation is essential. Fig. 2 illustrates the DNN architectures considered in this work. However, for conciseness, the time complexity evaluation process is only detailed for the MLP architecture.

To evaluate the time complexity of the MLP, first, we evaluate the main operations involved in a single-layer neural network, *i.e.*, the perceptron, and formulate these operations' time complexities. Then, based on the training and prediction algorithms, we evaluate the whole MLP time complexity.

The main operations in the perceptron are the feed-forward, error computation, and weights updates. Considering the perceptron illustrated in Fig. 3, these operations are defined in Table 1.

**TABLE 1. Perceptron training operations.**

| Operation | Equation |
|---|---|
| 1. Feed-forward | $o = \sum_{i=0}^{n} w_i x_i$ |
| 2. Error computation | $\Delta w_i = \eta(t - o)x_i$ |
| 3. Weights updates | $w_i = w_i + \Delta w_i$ |

In these definitions, $x_0$ represents the bias unit. The input values (features) are represented by $x_1, x_2, \ldots, x_n$. The weights are represented by $w_i$, and $o$ represents the computed output unit. The learning rate is represented by $\eta$, and the targeted output is represented by $t$.

In time complexity analysis, operations that have constant times can be neglected because the goal is to highlight how

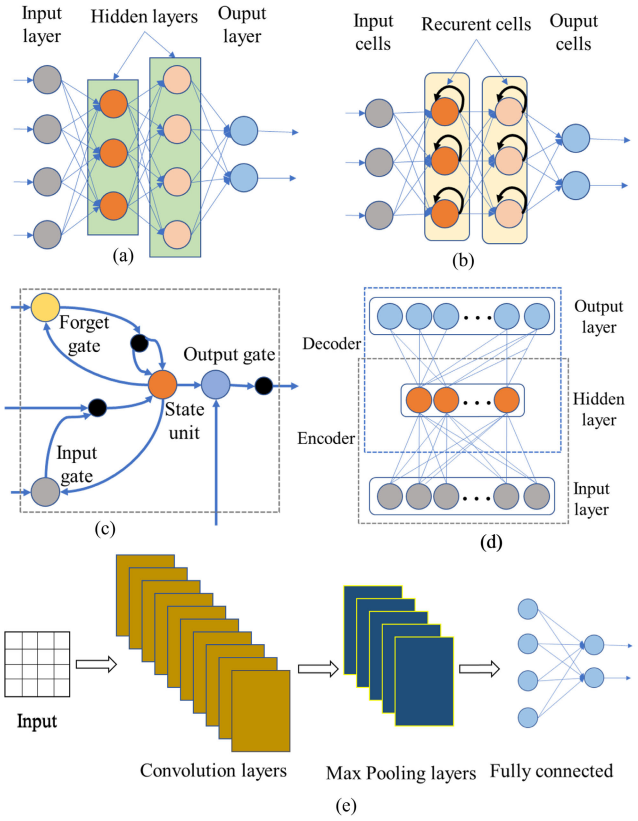


**FIGURE 2. The architecture of MLP(a), RNN(b), LSTM(c), Auto Encoder(d), and CNN(e).**

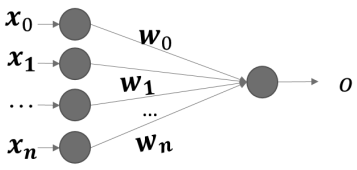


**FIGURE 3. Single-layer neural network: Perceptron.**

the latency scales with respect to the variables. Therefore, to simplify the time complexity approximation, we make the following assumptions:

- In the feed-forward, the time complexity to compute each perceptron (hidden or output unit) is identical and represented by $T_{FW}$;
- In the error computation, the time complexity to compute each output unit error is identical and identical by $T_{OE}$;
- In the weights update, the time complexity to update each weight is identical and represented by $T_{WU}$;

The neural network training consists of recursive feed-forward, error computation, and back-propagation to adjust the network weights. The condition of the recursion is to get the values of weights that minimize the output error on a particular set of training data. Algorithm 1 defines the MLP training process.

**Algorithm 1** MLP Training

**Result:** Weights that minimize the prediction error.
**Input 1:** *training_examples*, where each instance is represented by $\langle \vec{x}, t \rangle$, $\vec{x}$ represents the feature vector and $t$ is the class of the instance.
**Input 2:** Weights $w_{i,j}$ where $i$ and $j$ represent the input and output units, respectively.
**Input 3:** Learning rate $\eta$.
Initialize all weights $w_{i,j}$ to small random numbers;
**while** *termination condition not satisfied* **do**
  **foreach** $\langle \vec{x}, t \rangle$ *in training_examples* **do**
    Execute **Feed-forward**
    **foreach** *hidden and output unit j* **do**
      $o_j = \sum_{i \in inputs} w_{i,j}\, o_i$
    **end**
    Compute **Outputs errors**
    **foreach** *last layer unit k* **do**
      $\delta_k = o_k(1 - o_k)(t_k - o_k)$
    **end**
    Execute **Back-propagation**
    **foreach** *hidden unit h* **do**
      $\delta_h = o_h(1 - o_h) \sum_{k \in outputs} w_{h,k}\, \delta_k$
    **end**
    Execute **Weights update**
    **foreach** *network weight* $w_{i,j}$ **do**
      $\Delta w_{i,j} = \eta \delta_j x_{i,j}$
      $w_{i,j} = w_{i,j} + \Delta w_{i,j}$
    **end**
  **end**
**end**

**TABLE 2.** Time complexity for MLP training operations.

| Operation | Time complexity |
|---|---|
| Feed-forward | $C_{O1} = N_{nu} T_{FW}$ |
| Output Error | $C_{O2} = N_{out} T_{OE}$ |
| Back-propagation | $C_{O3} = (\sum N_{h_i}) T_{BP}$ |
| Weights updates | $C_{O4} = N_{wg} T_{WU}$ |

The time complexity of this training depends on the following variables:

- $N_{in}$ = the number of input units (feature vector size),
- $N_{h_i}$ = the number of units of the hidden layer $h_i$,
- $N_{out}$ = the number of output units (number of classes),
- $N_{nu}$ = the total number of units (perceptrons),
- $N_{wg}$ = the total number of weights,
- $N_{epoch}$ = the number of iterations in the training dataset,
- $N_{te}$ = the number of training examples,

where: $N_{nu}$ and $N_{wg}$ are defined in (1) and (2), respectively.

$$N_{nu} = N_{in} + \sum N_{h_i} + N_{out} \tag{1}$$

$$N_{wg} = N_{in} N_{h_1} + \sum N_{h_i} N_{h_{i+1}} + N_{h_n} N_{out} \tag{2}$$

To simplify the time complexity approximation, we also assume in the back-propagation that the time complexity to compute each hidden unit is identical and represented by $T_{BP}$.

**TABLE 3.** Training and prediction latency on different MLP and dataset sizes.

| Model | M1 | M2 | M3 | M4 | M5 |
|---|---|---|---|---|---|
| $N_{te}$ | 200k | 200k | 200k | 200k | 200k |
| $N_{in}$ | 100 | 90 | 80 | 70 | 60 |
| $h_1$ | 150 | 100 | 100 | 50 | 50 |
| $h_2$ | 150 | 100 | 50 | 50 | 50 |
| $h_3$ | 150 | 100 | 100 | 100 | 50 |
| $N_{out}$ | 50 | 50 | 10 | 10 | 1 |
| $N_{epoch}$ | 30 | 25 | 20 | 15 | 10 |
| $C(tr)$ | **36.5 mn** | **15.8 mn** | **7.4 mn** | **3.7 mn** | **1.7 mn** |
| $C(pr)$ | **201 ms** | **147 ms** | **114 ms** | **94 ms** | **71 ms** |

Table 2 presents the time complexity of each operation for one iteration in the training. The time complexity for training the MLP is defined in (3). Using the trained MLP network, the prediction phase is only as complex as the feed-forward operations. The time complexity of this phase is defined in (4).

$$\begin{aligned} C(training) &= N_{epoch} N_{te} [C_{O1} + C_{O2} + C_{O3} + C_{O4}] \\ &= N_{epoch} N_{te} [N_{nu} T_{FW} + N_{out} T_{OE} \\ &\quad + (\sum N_{h_i}) T_{BP} + N_{wg} T_{WU}] \\ &= N_{epoch} N_{te} [(N_{in} + \sum N_{h_i} + N_{out}) T_{FW} \\ &\quad + N_{out} T_{OE} + (\sum N_{h_i}) T_{BP} \\ &\quad + (N_{in} N_{h_1} + \sum N_{h_i} N_{h_{i+1}} \\ &\quad + N_{h_n} N_{out}) T_{WU}] \end{aligned} \tag{3}$$

$$\begin{aligned} C(prediction) &= C_{O1} \\ &= N_{nu} T_{FW} \\ &= (N_{in} + \sum N_{h_i} + N_{out}) T_{FW} \end{aligned} \tag{4}$$

To better visualize how the DNN structure affects the latency, we have generated simulation data containing 100 features, 50 attacks categorized into 10 attack categories, and 200,000 (200k) training examples. This simulation dataset is used solely to illustrate the latency evolution in the training and prediction of models that use DNNs with various structures. To evaluate and compare the models' accuracy, more data are used in the experimentation section. Using the simulation dataset, we trained and evaluated five MLP models of various sizes (M1 to M5). Table 3 describes these models and presents the training latency expressed in minutes (mn) and the prediction latency expressed in milliseconds (ms). Fig. 4 illustrates the latency evolution according to the MLP and data sizes. The same process is applied to evaluate the time complexity of all the other DNNs, *i.e.*, RNN, LSTM, Auto Encoder, and CNN.

In this analysis, it is crucial to point out that the size of the DNNs and consequently their latencies are also determined by the characteristics of the data to be classified, *i.e.*, feature vector size, number of classes, and data distribution. Firstly, the number of features and classes preset the number of neurons in the input and output layers of the DNN, respectively. Secondly, the more complex the data distribution is, the more

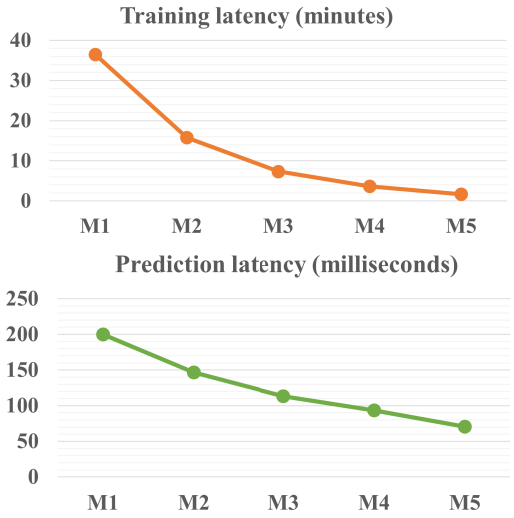


**FIGURE 4.** Training and prediction latency according to MLP and dataset sizes.

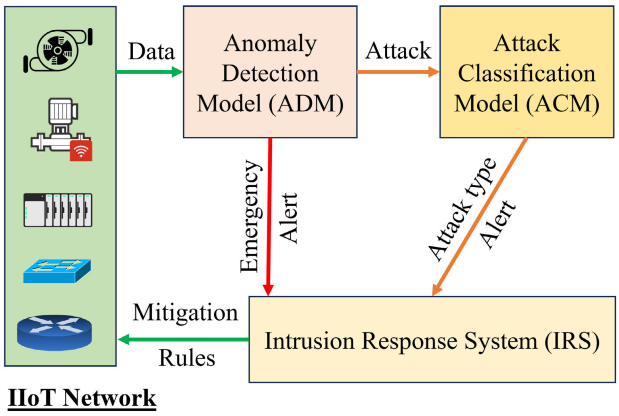


**FIGURE 5.** General architecture of our collaborative IDPS framework for ICSs.

hidden layers and neurons at each hidden layer are needed in order to shape the decision boundaries. Furthermore, the more complex this data distribution is, the more training examples and iterations on the training examples (number of epochs) are needed.

## III. PROPOSED COLLABORATIVE IDS AND DEPLOYMENT ARCHITECTURE

As demonstrated in the previous section, IDSs that employ complex neural networks yield high training and prediction latency. Yet, complex data classification, such as IIoT datasets, generally requires complex DNNs. Specifically, the more classes ($N_{out}$) we have, the more features ($N_{in}$), training examples ($N_{te}$), iterations in the training examples ($N_{epoch}$), and layers in the DNNs ($N_h$) are needed to find the best decision boundaries. However, in mission-critical ICSs, intrusion detection requires very low latency. Therefore, to reduce the latency in the proposed IDPS, we employ a technique that we previously designed for heavy ensemble learners [17], [18] and design a classification scheme that splits the problem into two collaborative tasks: anomaly detection and attack classification. Fig. 5 illustrates the architecture of our collaborative IDPS framework for ICSs.

By performing anomaly detection as a first task, the proposed collaborative IDPS can meet mission-critical IIoT latency requirements. This latency reduction is achieved by reducing most of the variables that impact the time complexity of the DNNs. Firstly, as a binary classification, anomaly detection has fewer neurons at the output layer ($N_{out} = 2$). Secondly, having a less complex decision function than a multi-class classification, binary classification could reach an optimal accuracy with fewer $N_h$, $N_{h_i}$, and $N_{epoch}$. Moreover, binary classification could require fewer features than multi-class classification. Therefore, anomaly detection can benefit from more dimensionality reduction of the features, *i.e.*, reduced $N_{in}$, (*e.g.*, M5 presented in section II). Finally, being less computationally intensive than a multi-class classification, the anomaly detection model can be deployed in any local server in the ICS stations and benefit from less communication latency. When an anomaly is detected, the information is immediately sent to the IRS and the security administrators to apply urgent and emergency response measures. Then, the anomalous traffic is sent to an attack classification model.

The attack classification task is less latency-sensitive than anomaly detection. Therefore, this classification model can employ complex DNNs and be deployed on servers with more computation resources, such as cloud servers. In contrast to binary classification, attack classification could require more features. Thus, for this model, more classification features can be included during the training and prediction to improve the accuracy. When an attack class is predicted, the information is sent to the IRS and the security administrators to apply complementary and more precise prevention mechanisms according to the attack type.

The deployment architecture of our proposed IDPS is illustrated in Fig. 6. This architecture adopts the innovative SDN and Network Functions Virtualization (NFV) to efficiently implement the IDPS in IIoT networks. In every station of the IIoT network, the sensors and actuators are connected to a PLC through an SDN switch. The data from the switch is collected by a local server, where anomaly detection is performed. A station may be connected to remote stations through various wide-area network protocols. SDN and NFV enable on-demand provisioning of the network functions, including the IDS and IRS [19]. For the IDS, SDN enables on-demand monitoring of the resources, where the set of resources to be monitored can be dynamically changed [20], [21]. In addition, the IDS performance can be improved by advanced monitoring features. For the response module, SDN architectures enable automatic and real-time reactions to block or redirect malicious traffic [22]. Moreover, the network's programmability allows management automation

and minimizes human intervention and related operational costs. Fig. 7 details how the IDPS modules operate using the Unified Modeling Language (UML) sequence diagram. This diagram depicts the interaction between the main entities of the IDPS in the order in which these interactions take place. The main entities include the data collection gates (SDN switches), the system orchestrator, the Anomaly Detection Model (ADM), the Attack Classification Model (ACM), the IRS, and the SDN controller.

This collaborative IDPS provides a faster intrusion detection system with attack classification for efficient response. In this architecture, only anomalous traffic needs to go through the complex attack classification DNN, which represents an extremely small portion of the IIoT data. The next section presents the detection features selection, the implemented deep learning models, and the response measures.

## IV. PROPOSED DETECTION FEATURES, LEARNING MODEL, AND IRS MEASURES

### A. PROPOSED DETECTION FEATURES

The choice of the feature set is challenging in every machine learning solution. It is more convenient in some applications to gather all the features that are available (the initial feature set), then use dimensionality reduction methods to select the best features (the final feature set). For instance, with image processing, it is feasible to take all pixels instead of studying and identifying each relevant feature. However, an initial feature set is not directly apparent in some applications. In intrusion detection for IIoT applications, feature selection is a more complex task regarding the multiplicity and variety of elements involved in cyber-physical systems (functional organization, software, operational principles, network workload, protocols, hardware, power consumption, etc.). Thus, it is crucial to study and identify relevant features and constitute an initial feature set [23]. Then, the dimensionality reduction methods can be applied to keep the best features. In this study, we select the detection features according to the attacks' behavior and the IIoT network's characteristics. Algorithm 2 describes the methodology adopted for feature selection. This is a complex task but convenient as it can clearly explain the role of each feature used in the IDS. Table 4 presents all the proposed detection features and their roles.

**Algorithm 2** IIoT Features Selection Logic

**Output:** Relevant feature set (Relevant_Features).
**Input 1:** Targeted IIoT cyber-threats (Targeted_cyber_threats).
**Input 2:** All IIoT system features (All_IIoT_Features).
**foreach** $attack_x$ *in Targeted_cyber_threats* **do**
  **foreach** $feature_y$ *in All_IIoT_Features* **do**
    **if** $feature_y$ *is changed by* $attack_x$ **then**
      Add $feature_y$ in Relevant_Features
    **end**
  **end**
**end**
Delete redundancy in Relevant_Features
Delete correlation in Relevant_Features
Reduce dimensionality of Relevant_Features

**TABLE 4. Relevant features for IIoT IDSs.**

| Features | Definition | Role |
|---|---|---|
| Time-based traffic features | Statistics from TCP/IP traffic flows using a predefined time intervals | Detection of Scanning, Probing, and DoS attacks that operate with many connections in short time intervals |
| Connection-based traffic features | Statistics from TCP/IP traffic flows based on a window of a predefined number of connections [23] | Detection of Scanning, Probing, and DoS attacks that employ large time intervals |
| Packet header features | Metadata portions of TCP/IP packets, such as IP addresses, port numbers, services, and flags [24] | Detection of SYN flood, Man-In-The-Middle (MITM), and ICMP/UDP/TCP fragmentation attacks |
| Content features | Information contained in the TCP/IP packet payload [25]–[29], *e.g.*, number of failed login attempts | Detection of Remote to Local (R2L) and User to Root (U2R) attacks |
| Wireless communication features | Radio signal characteristics, such as Received Signal Strength (RSS), Signal-to-noise ratio (SNR), distance of the radio transmitter, Radio-frequency fingerprint (RFF) [30]–[32], [32] | Detection of attacks that involve wireless devices, including jamming, de-authentication, spoofing, and contamination |

### B. PROPOSED DETECTION MODELS

Multiple DNN algorithms have demonstrated excellent capabilities in building accurate models for different real-life problems. In spite of this, there is no specific DNN that will produce the most accurate learner for every dataset. Each DNN focuses on particular aspects of the data distribution to build the models. For example, CNNs learn spatial relations between separate features and perform at their best when dealing with data with a grid-like topology, such as images. Meanwhile, RNNs and LSTMs are specialized in learning temporal or sequential information and are more suitable for temporal data, such as text and speech analysis. In intrusion detection, each DNN focuses on some part of the data patterns. Therefore, it is necessary to implement and evaluate different DNNs and deploy only those that produce the best detection models for the specified data. Thus, this work studies and implements the most promising DNNs, including

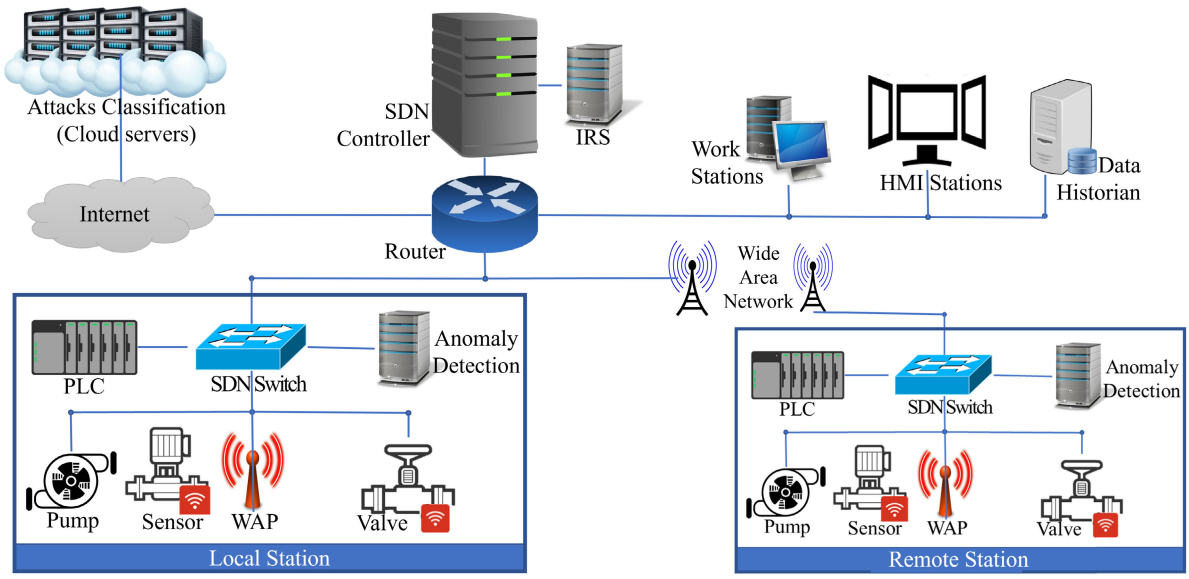


**FIGURE 6.** The proposed IDPS architecture in the ICS.

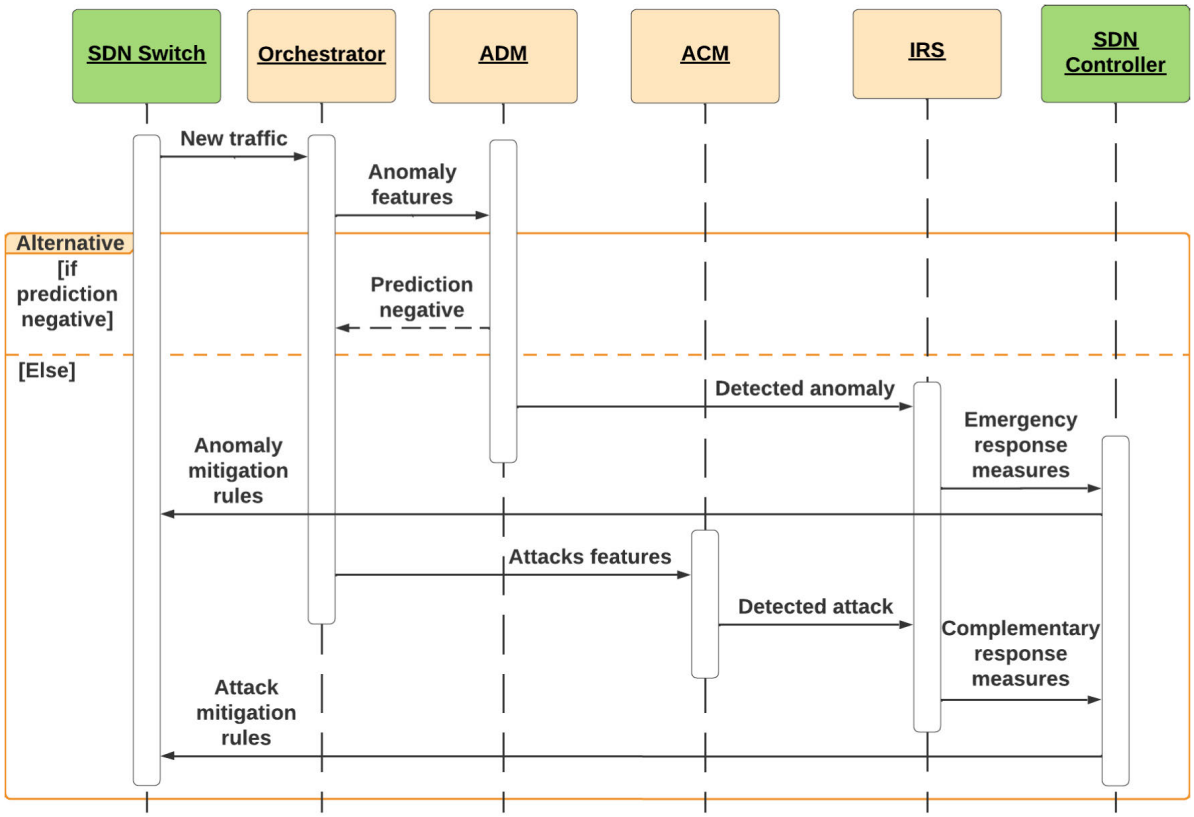


**FIGURE 7.** The UML sequence diagram of the proposed IDPS.

CNN, RNN, LSTM, and MLP. The anomaly detection model may employ a simple DNN, such as MLP; however, the attack classification model uses a more complex DNN, such as RNN, LSTM, and CNN.

### C. PROPOSED RESPONSE MEASURES

The success of some attacks depends mainly on the time gap between the detection and the defensive response against the attack. Intrusion Response Systems (IRSs), also referred to as

Intrusion Prevention Systems (IPSs), launch counteractions automatically when attacks are detected by the IDS, in order to defend the targeted system. Compared to IDSs that just generate reports or alarms, IRSs reduce the vulnerability window between when an intrusion is detected and when defensive actions are taken. Therefore, this work proposes an emergency and quasi-real-time response when an anomaly is detected. Then, it classifies this anomaly to identify suitable complementary measures according to the attack type.

The basic mitigation procedure is to block the packets involved in the reported intrusion and reject all incoming and outgoing traffic of the malicious device (*e.g.*, using blocklist or allowlist MAC filtering). This can stop many attacks, including R2L and U2R, and safeguard the ICS's confidentiality and integrity. This emergency measure can also maintain the system's services by reducing scanning, probing, and DoS attacks. It is also possible to deflect and redirect the malicious users into a Honeypot or other attack analysis systems to gain more information on the way they are operating [33], [34]. These preventative measures are implemented in the SDN-based deployment architecture. For instance, when the IRS receives an anomaly message from the ADM, the IRS uses the information contained in the received message to build the set of SDN rules and send it to the SDN controller (Fig. 7). Then, the controller transmits those rules to the forwarding devices' flow tables to drop the concerned packets or forward them to an attack analysis system. Likewise, when the IRS receives an attack message from the ACM, it uses this information to build the set of SDN rules and send it to the controller. The SDN controller transmits this routing information to the forwarding devices' flow tables to mitigate the attack. For further response, it is possible to develop an additional module that can counterattack the attacking entity to neutralize or attenuate its impact.

The complementary response measures are defined according to each specific attack type. For example, to overcome a detected jamming attack, communication bandwidth can be switched to other frequencies. To realize that, Dynamic Channel Selection (DCS) can be implemented [35]. The DCS enables the wireless transmitters to monitor the interference level, and when it exceeds the predefined DCS threshold, the wireless transmitters stop operating on that channel. Then, the wireless access point (WAP) uses automatic channel selection to determine an alternative channel to switch the communication. Furthermore, the robustness of the legitimate signal can be increased in order to maintain secure wireless communication during jamming attacks. For advanced defence, tracking systems can be employed to locate the jamming station and terminate it [36]. These technologies are also effective against other active wireless attacks, including contamination and spoofing.

## V. EXPERIMENTATION

This section presents the experiment and discusses the results. The experimentation aims to evaluate the performance of the DNN-based collaborative IDS. Therefore, we exploit three datasets that present different challenges.

The first dataset, WUSTL-IIOT-2018, presented in [8] and [37], is used for SCADA cyber-security research. The dataset was built using a SCADA system test-bed. To generate this data, scan tools were used to inspect the topology of the victim network and identify the devices in the network, as well as their vulnerabilities. The attacks carried out against the test-bed include Port Scanner, Address Scan Attacks, Device Identification Attacks, and Exploit. All network traffic (normal and abnormal traffic) was monitored by the Audit Record Generation and Utilization System (ARGUS) tool. The traffic captured comprises 7,049,989 observations, with 93.93% being normal traffic (without attacks) and 6.07% being abnormal traffic (attacked traffic). The raw data has 25 networking features, where some features are used to classify the data, while others are used to train and test machine learning algorithms. After the data cleaning process and dimensionality reduction, a Comma-Separated Values (CSV) file containing 7,037,983 observations (vectors in the dataset) and five features were provided. Each vector in the dataset is labelled as normal or attack, depending on the case.

The second dataset, NSL-KDD, was built from a regular network (not an IIoT); however, it is one of the most complete, realistic, and challenging datasets available, which is used to compare machine learning-based IDSs [38]. NSL-KDD includes a total of 39 specific attacks regrouped into four different attack categories, namely, probing (Probe), DoS, R2L, and U2R. The training dataset (KDDTrain+) and testing dataset (KDDTest+) contain 125,973 and 22,544 single connection vectors, respectively. Each connection vector is represented by 41 features and labelled as either normal or attack, with exactly one specific attack type. This dataset is also available in a CSV file.

The third dataset, UNSW-NB15, presented in [39] and [40] was also built from a regular network; however, unlike the NSL-KDD dataset, it contains a hybrid of the modern normal and abnormal network traffic. To generate this data, the IXIA PerfectStorm tool was utilized in the cyber range lab of the Australian Centre for Cyber Security. UNSW-NB15 includes a total of 205 specific attacks regrouped into nine attack categories, namely, Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, and Worms. The dataset, provided in a CSV file, contains a total of 2,540,044 records, each represented by 47 features and labelled as either normal or attack, including the specific attack category.

The experimentation employs four different DNNs to build the detection models. These DNNs include MLP, Simple RNN, LSTM, and CNN. The implementation is conducted in the anomaly detection and the attacks classification stages. The machine learning platform used to train and evaluate the models is scikit-learn (sklearn), a free software library for the Python programming language, installed with Anaconda, an open-source distribution for Python and R.

### A. THE ANOMALY DETECTION MODEL

We begin the implementation with the anomaly detection model. Table 5 presents the distribution of the instances in the binary WUSTL-IIOT-2018, NSL-KDD, and UNSW-NB15 datasets.

**TABLE 5. Instances distribution in the binary WUSTL-IIOT-2018, NSL-KDD, and UNSW-NB15 datasets.**

| Class | Number of instances | | | |
|---|---|---|---|---|
| | *WUSTL-IIOT* | *KDDTrain+* | *KDDTest+* | *UNSW-NB15* |
| Normal | 6,610,778 | 67,343 | 9,711 | 2,218,761 |
| Attack | 427,205 | 58,630 | 12,833 | 321,283 |
| **Total** | **7,037,983** | **125,973** | **22,544** | **2,540,044** |

For each DNN model, we tuned the hyper-parameters, such as the number of hidden layers, number of units for each hidden layer, learning rates, activation and solver functions, number and size of the CNN filters, and number of epochs. For WUSTL-IIOT-2018 and UNSW-NB15, we split the dataset into training (80%) and testing (20%) sets. For the NSL-KDD, we trained the models using KDDTrain+ and evaluated them with KDDTest+. Fig. 8, Fig. 9, and Fig. 10 present the accuracy of the models on the WUSTL-IIOT-2018, NSL-KDD, and UNSW-NB15 datasets, respectively.

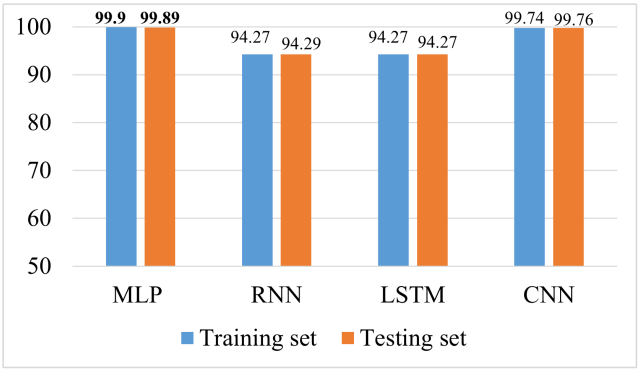


**FIGURE 8. Accuracy of anomaly detection models using binary-labelled WUSTL-IIOT-2018 dataset.**

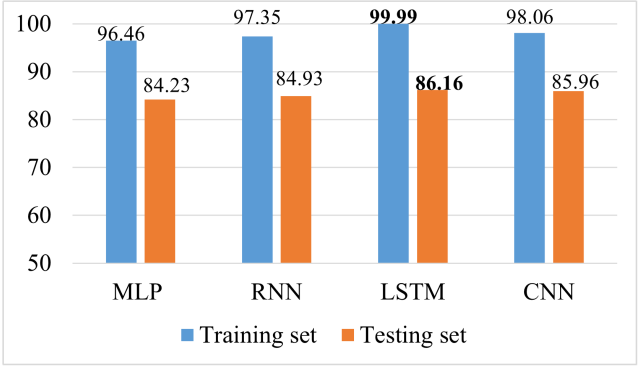


**FIGURE 9. Accuracy of anomaly detection models using binary-labelled NSL-KDD dataset.**

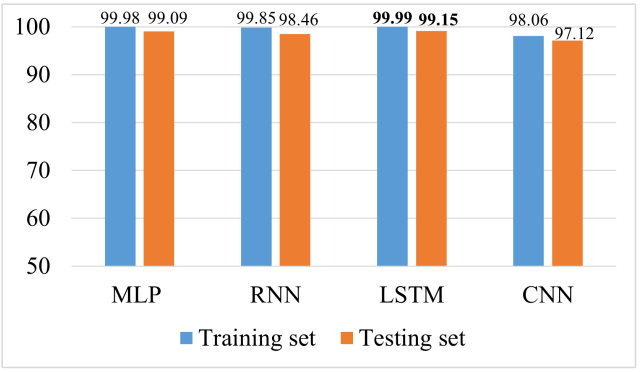


**FIGURE 10. Accuracy of anomaly detection models using binary-labelled UNSW-NB15 dataset.**

For each model, the accuracy presented is obtained from tuned hyper-parameters.

The experimentation results with the WUSTL-IIOT-2018 and UNSW-NB15 datasets confirm that anomaly detection does not require complex DNNs to produce a high-accuracy prediction model. In both datasets, lightweight DNNs (*e.g.*, MLP with one hidden layer of 100 neurons) were able to reach over 99% detection accuracy. Such DNNs can be easily trained and deployed on a local server for ultra-low latency intrusion detection and response. In our local server, the training time takes less than 15 minutes and 6 minutes for WUSTL-IIOT-2018 and UNSW-NB15, respectively. The prediction time for the test datasets takes approximately 0.7 seconds (WUSTL-IIOT-2018) and 0.4 seconds (UNSW-NB15). However, for the NSL-KDD dataset, the models show a lower accuracy (less than 87%). This can be explained by the fact that this dataset was built deliberately from hard-to-classify samples. Therefore, the classification in this dataset presents more challenges compared to other datasets.

### B. THE ATTACK CLASSIFICATION MODEL

The second part of the experimentation concerns the attacks classification model. Since the published WUSTL-IIOT-2018 dataset does not indicate the attack types, we only use the NSL-KDD and UNSW-NB15 datasets. Table 6 and Table 7 present these datasets' organization for the multi-class classification.

**TABLE 6. Attacks instances distribution in the NSL-KDD dataset.**

| Class | Number of instances | |
|---|---|---|
| | **KDDTrain+** | *KDDTest+* |
| DOS | 45,927 | 7,458 |
| Probe | 11,656 | 2,421 |
| R2L | 995 | 2,754 |
| U2R | 52 | 200 |
| **Total** | **58,630** | **12,833** |

The same DNNs, *i.e.*, MLP, Simple RNN, LSTM, and CNN, are used to build the attack classification models. Fig. 11 and Fig. 12 illustrate the accuracy of the models on NSL-KDD and UNSW-NB15 datasets, respectively. For

**TABLE 7.** Attacks instances distribution in the UNSW-NB15 dataset.

| Class | Number of instances |
|---|---|
| Fuzzers | 24,246 |
| Analysis | 2,677 |
| Backdoors | 2,329 |
| DoS | 16,353 |
| Exploits | 44,525 |
| Generic | 215,481 |
| Reconnaissance | 13,987 |
| Shellcode | 1,511 |
| Worms | 174 |
| **Total** | **321,283** |

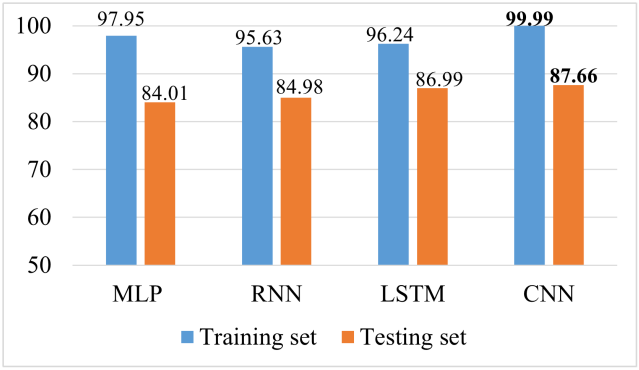


**FIGURE 11.** Accuracy of attack classification models using categorized-labelled NSL-KDD dataset.

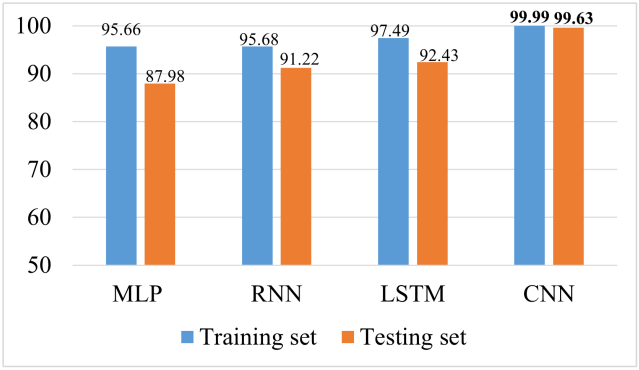


**FIGURE 12.** Accuracy of attack classification models using categorized-labelled UNSW-NB15 dataset.

every model, the presented accuracy is obtained from the tuned hyper-parameters.

This experiment confirms that the attack classification may require complex DNNs and a long training time. With NSL-KDD, the best accuracy of 87.66% was provided by the CNN model, followed by the LSTM model with 86.99% accuracy. Compared to previous works that employed the NSL-KDD dataset, the proposed collaborative DNN-based IDS shows better prediction accuracy. Table 8 compares these previous solutions with our attack classification model (Four-category classification). With UNSW-NB15, the best accuracy of 99.63% was also provided by the CNN model, followed by the LSTM model with 92.43% accuracy.

**TABLE 8.** Comparison with solutions that exploits the NSL-KDD dataset for multi-class classification.

| Methods used | Accuracy |
|---|---|
| Proposed collaborative deep learning-based IDS | 87.66 % |
| Methods in [38]:<br>- NBTree<br>- Random Tree<br>- Decision Trees J48<br>- Random Forest<br>- MLP | -<br>82.02 %<br>81.59 %<br>81.05 %<br>80.67 %<br>77.41 % |
| Methods in [41]:<br>- SimpleCart<br>- MLP | -<br>82.32 %<br>73.54 % |
| RNNs [42]. | 81.29 % |
| ANN with tansig transfer function,Levenberg-Marquardt (LM) and BFGS quasi-Newton Back-propagation [43] | 81.20 % |
| A Two-Layer Dimension Reduction and Two-Tier Classification Model [44] | 84.86 % |
| Two-stage intrusion detection technique combining Naive Bayes and k-means [45] | 86.46 % |

**TABLE 9.** Comparison with solutions that exploits the UNSW-NB15 dataset for multi-class classification.

| Methods used | Accuracy |
|---|---|
| Proposed collaborative deep learning-based IDS | 99.63 % |
| Methods in [40]:<br>- Decision Tree<br>- Logistic Regression<br>- Naïve Bayes<br>- Artificial Neural Network<br>- Expectation-Maximisation Clustering | -<br>85.56 %<br>83.15 %<br>82.07 %<br>81.34 %<br>78.47 % |
| Random Forest using different subsets of features [46]:<br>- Subset 2<br>- Subset 1 | -<br>81.61%<br>75.66 % |
| Deep neural network models in [47]:<br>- Dense-layer<br>- LSTM | -<br>79.34%<br>79.21 % |
| Extreme gradient boosting (XGBoost) [48] | 75.88 % |
| Support Vector Machine [49] | 75.77 % |

Compared to previous works that employed the UNSW-NB15 dataset, the proposed collaborative DNN-based IDS shows again better prediction accuracy. Table 9 compares these previous solutions with our attack classification model (nine-category classification). In our experimentation environment, the training of the attack classification models can take several hours, and the prediction takes approximately 10 seconds and 3 minutes 36 seconds for NSL-KDD and UNSW-NB15 test datasets, respectively. However, as this task can be done in the cloud, the complexity can be handled thanks to the availability of more computation resources.

This experiment demonstrates that this approach is efficient for intrusion detection in mission-critical smart factories. The anomaly detection is performed locally with a lightweight DNN, which offers an ultra-low latency IDS for fast response in the IRS. Then, a larger DNN deployed in the cloud provides a robust attack classification for more precise responses in the IRS, *i.e.*, responses that launch adequate mitigation measures according to the category of the detected attacks.

## VI. CONCLUSION

A deep learning-based IDS produces better predictions for future networks, but the more complex the neural network structure, the greater its impact on latency. To leverage these innovative learning models and deal with the latency requirement of ICSs, this work introduced a time complexity analysis of the DNN algorithms to highlight the variables with the most impact on the training and prediction latency. Based on this analysis, a collaborative DNN-based IDPS that employs two classification models is proposed. The first model employs a lightweight DNN that performs low latency anomaly detection, *i.e.*, a simple binary classification. This lightweight DNN is deployed on local servers to allow faster threat detection and emergency response. The second model performs attack classifications of the anomalous traffic to guide the intrusion response tasks. This second classifier can be deployed in the cloud to benefit from more computation resources to run more complex DNNs.

Moreover, an SDN-based deployment architecture of the proposed collaborative IDPS in ICS networks was presented. This architecture provided an efficient implementation of the IDPS with key innovative features, such as on-demand resources monitoring on the IDS and real-time response on the IRS. Furthermore, this work proposed various detection features, response measures, and implemented different learning methods, including CNN, LSTM, RNN, and MLP. The experimentation, which was performed on three datasets with different challenges, demonstrated the efficiency of the proposed approach. This can be further improved by investigating more DNNs in our future research.

## REFERENCES

[1] F. Stoessel, *Thermal Safety of Chemical Processes: Risk Assessment and Process Design*. Hoboken, NJ, USA: Wiley, 2021.

[2] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human–robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, 2017.

[3] M. Zhao and M. Barati, "Substation safety awareness intelligent model: Fast personal protective equipment detection using GNN approach," *IEEE Trans. Ind. Appl.*, vol. 59, no. 3, pp. 3142–3150, May/Jun. 2023.

[4] Statista Research Department. (2021). *Number of Explosion Incidents in the United States from 2012 to 2020*. Accessed: Aug. 2021. [Online]. Available: https://www.statista.com/statistics/785950/number-of-explosion-incidents-in-the-united-states/

[5] S. Al-Rabiaah, "The 'Stuxnet' virus of 2010 as an example of a 'APT' and its 'recent' variances," in *Proc. 21st Saudi Comput. Soc. Nat. Comput. Conf. (NCC)*, Apr. 2018, pp. 1–5.

[6] J. M. Beaver, R. C. Borges-Hink, and M. A. Buckner, "An evaluation of machine learning methods to detect malicious SCADA communications," in *Proc. 12th Int. Conf. Mach. Learn. Appl. (ICMLA)*, vol. 2, Dec. 2013, pp. 54–59, Dec. 2013.

[7] C.-T. Lin, S.-L. Wu, and M.-L. Lee, "Cyber attack and defense on industry control systems," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 524–526.

[8] M. Teixeira, T. Salman, M. Zolanvari, R. Jain, N. Meskin, and M. Samaka, "SCADA system testbed for cybersecurity research using machine learning approach," *Future Internet*, vol. 10, no. 8, p. 76, Aug. 2018.

[9] I. Ullah and Q. H. Mahmoud, "A hybrid model for anomaly-based intrusion detection in SCADA networks," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2017, pp. 2160–2167.

[10] A. Le, P. Dinh, H. Le, and N. C. Tran, "Flexible network-based intrusion detection and prevention system on software-defined networks," in *Proc. Int. Conf. Adv. Comput. Appl. (ACOMP)*, Nov. 2015, pp. 106–111.

[11] A. Al-Abassi, H. Karimipour, A. Dehghantanha, and R. M. Parizi, "An ensemble deep learning-based cyber-attack detection in industrial control system," *IEEE Access*, vol. 8, pp. 83965–83973, 2020.

[12] B. Li, Y. Wu, J. Song, R. Lu, T. Li, and L. Zhao, "DeepFed: Federated deep learning for intrusion detection in industrial cyber–physical systems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 8, pp. 5615–5624, Aug. 2021.

[13] J. Ling, Z. Zhu, Y. Luo, and H. Wang, "An intrusion detection method for industrial control systems based on bidirectional simple recurrent unit," *Comput. Electr. Eng.*, vol. 91, May 2021, Art. no. 107049.

[14] R. F. Mansour, "Artificial intelligence based optimization with deep learning model for blockchain enabled intrusion detection in CPS environment," *Sci. Rep.*, vol. 12, no. 1, p. 12937, Jul. 2022.

[15] Q. Shafi, A. Basit, S. Qaisar, A. Koay, and I. Welch, "Fog-assisted SDN controlled framework for enduring anomaly detection in an IoT network," *IEEE Access*, vol. 6, pp. 73713–73723, 2018.

[16] O. Alkadi, N. Moustafa, B. Turnbull, and K. R. Choo, "A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks," *IEEE Internet Things J.*, vol. 8, no. 12, pp. 9463–9472, Jun. 2021.

[17] P. Illy, G. Kaddoum, C. Miranda Moreira, K. Kaur, and S. Garg, "Securing fog-to-things environment using intrusion detection system based on ensemble learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2019, pp. 1–7.

[18] P. Illy, G. Kaddoum, P. F. de Araujo-Filho, K. Kaur, and S. Garg, "A hybrid multistage DNN-based collaborative IDPS for high-risk smart factory networks," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 4273–4283, Dec. 2022.

[19] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, 2015.

[20] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "An SDN-supported collaborative approach for DDoS flooding detection and containment," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2015, pp. 659–664.

[21] A. Hermosilla, A. M. Zarca, J. B. Bernabe, J. Ortiz, and A. Skarmeta, "Security orchestration and enforcement in NFV/SDN-aware UAV deployments," *IEEE Access*, vol. 8, pp. 131779–131795, 2020.

[22] M. Campos and J. S. B. Martins. (2017). *A SDN-Based Flexible System for On-the-Fly Monitoring and Treatment of Security Events*. [Online]. Available: https://zenodo.org/record/1291094

[23] P. Illy, G. Kaddoum, K. Kaur, and S. Garg, "ML-based IDPS enhancement with complementary features for home IoT networks," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 2, pp. 772–783, Jun. 2022.

[24] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, "A supervised intrusion detection system for smart home IoT devices," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019.

[25] W. Zhong, N. Yu, and C. Ai, "Applying big data based deep learning system to intrusion detection," *Big Data Mining Anal.*, vol. 3, no. 3, pp. 181–195, Sep. 2020.

[26] A. Abusitta, M. Bellaiche, M. Dagenais, and T. Halabi, "A deep learning approach for proactive multi-cloud cooperative intrusion detection system," *Future Gener. Comput. Syst.*, vol. 98, pp. 308–318, Sep. 2019.

[27] F. Iglesias and T. Zseby, "Analysis of network traffic features for anomaly detection," *Mach. Learn.*, vol. 101, nos. 1–3, pp. 59–84, Oct. 2015.

[28] C. Xu, S. Chen, J. Su, S. M. Yiu, and L. C. K. Hui, "A survey on regular expression matching for deep packet inspection: Applications, algorithms, and hardware platforms," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2991–3029, 4th Quart., 2016.

[29] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48.

[30] B. W. Ramsey, B. E. Mullins, M. A. Temple, and M. R. Grimaila, "Wireless intrusion detection and device fingerprinting through preamble manipulation," *IEEE Trans. Dependable Secure Comput.*, vol. 12, no. 5, pp. 585–596, Sep. 2015.

[31] N. Soltanieh, Y. Norouzi, Y. Yang, and N. C. Karmakar, "A review of radio frequency fingerprinting techniques," *IEEE J. Radio Freq. Identificat.*, vol. 4, no. 3, pp. 222–233, Sep. 2020.

[32] Q. Tian, Y. Lin, X. Guo, J. Wen, Y. Fang, J. Rodriguez, and S. Mumtaz, "New security mechanisms of high-reliability IoT communication based on radio frequency fingerprint," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7980–7987, Oct. 2019.

[33] W. Tian, X. Ji, W. Liu, G. Liu, J. Zhai, Y. Dai, and S. Huang, "Prospect theoretic study of honeypot defense against advanced persistent threats in power grid," *IEEE Access*, vol. 8, pp. 64075–64085, 2020.
[34] I. M. M. Matin and B. Rahardjo, "The use of honeypot in machine learning based on malware detection: A review," in *Proc. 8th Int. Conf. Cyber IT Service Manage. (CITSM)*, Oct. 2020, pp. 1–6.
[35] V. Navda, A. Bohra, S. Ganguly, and D. Rubenstein, "Using channel hopping to increase 802.11 resilience to jamming attacks," in *Proc. IEEE 26th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, May 2007, pp. 2526–2530.
[36] W. Aldosari, M. Zohdy, and R. Olawoyin, "Tracking the mobile jammer in wireless sensor networks using extended Kalman filter," in *Proc. IEEE 10th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf. (UEMCON)*, Oct. 2019, pp. 207–212.
[37] S. Alem, D. Espes, E. Martin, L. Nana, and F. de Lamotte, "New dataset for Industry 4.0 to address the change in threat landscape," in *Proc. 15th Int. Conf. Risks Secur. Internet Syst. (CRiSIS)*. Paris, France: Springer, Nov. 2021, pp. 273–288.
[38] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl. (CISDA)*, Jul. 2009, pp. 1–6.
[39] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.
[40] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., A Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, Apr. 2016.
[41] K. Bajaj and A. Arora, "Improving the intrusion detection using discriminative machine learning approach and improve the time complexity by data mining feature selection methods," *Int. J. Comput. Appl.*, vol. 76, no. 1, pp. 5–11, Aug. 2013.
[42] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.
[43] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst. (SPACES)*, Jan. 2015, pp. 92–96.
[44] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K. R. Choo, "A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 7, no. 2, pp. 314–323, Apr. 2019.
[45] M. Vishwakarma and N. Kesswani, "A two-stage intrusion detection system (TIDS) for Internet of Things," in *Advances in Deep Learning, Artificial Intelligence and Robotics*. Cham, Switzerland: Springer, 2022, pp. 89–97.
[46] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *Proc. IEEE 26th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2017, pp. 1881–1886.
[47] M. Cavojský, G. Bugár, and D. Levický, "Comparative analysis of feedforward and RNN models for intrusion detection in data network security with UNSW-NB15 dataset," in *Proc. 33rd Int. Conf. Radioelektronika (RADIOELEKTRONIKA)*, Apr. 2023, pp. 1–6.
[48] A. Husain, A. Salem, C. Jim, and G. Dimitoglou, "Development of an efficient network intrusion detection model using extreme gradient boosting (XGBoost) on the UNSW-NB15 dataset," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2019, pp. 1–7.
[49] D. Jing and H.-B. Chen, "SVM based network intrusion detection for the UNSW-NB15 dataset," in *Proc. IEEE 13th Int. Conf. ASIC (ASICON)*, Oct. 2019, pp. 1–4.

**POULMANOGO ILLY** received the bachelor's degree in computer engineering, option analysis and programming, from École Supérieure d'Informatique (ESI), Université Nazi BONI (ex Université Polytechnique de Bobo-Dioulasso), Burkina-Faso, in 2014, and the M.S. degree in datascale (data management in large-scale distributed systems) from Université Paris Saclay-Université de Versailles Saint-Quentin-en-Yvelines, France, in 2017. He is currently pursuing the Ph.D. degree in electrical engineering with École de Technologie Supérieure (ÉTS), Université du Quebec, Montreal, Canada. His research findings are published in prestigious venues, such as IEEE WCNC, IEEE ISNCC, and IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT. His research interests include network security, the IoT security, intrusion detection, intrusion prevention, machine learning, and deep learning.

**GEORGES KADDOUM** (Senior Member, IEEE) received the bachelor's degree in electrical engineering from École Nationale Supérieure de Techniques Avancées (ENSTA Bretagne), Brest, France, in 2004, the M.S. degree in telecommunications and signal processing(circuits, systems, and signal processing) from Université de Bretagne Occidentale and Telecom Bretagne (ENSTB), Brest, in 2005, and the Ph.D. degree (Hons.) in signal processing and telecommunications from the National Institute of Applied Sciences (INSA), University of Toulouse, Toulouse, France, in 2009. Since 2010, he has been a Scientific Consultant in the field of space and wireless telecommunications for several U.S. and Canadian companies. In 2014, he was awarded the ÉTS Research Chair in physical-layer security for wireless networks. He is currently a Professor and a Tier 2 Canada Research Chair with École de Technologie Supérieure (ÉTS), Université du Québec, Montreal, Canada, and also a Faculty Fellow with the Cyber Security Systems and Applied AI Research Center, Lebanese American University, Beirut, Lebanon. He has published over 200 journal articles and conference papers and has two pending patents. His recent research activities cover mobile communication systems, modulations, security, and space communications and navigation. He received the Best Paper Award from the 2014 IEEE International Conference on Wireless and Mobile Computing, Networking, Communications (WIMOB), with three coauthors, and the 2017 IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC), with four coauthors. Moreover, he received the IEEE TRANSACTIONS ON COMMUNICATIONS Exemplary Reviewer Award, in 2015, 2017, and 2019. He received the Research Excellence Award from Université du Québec, in 2018. In 2019, he received the Research Excellence Award from ÉTS in recognition of his outstanding research outcomes. He is also serving as an Associate Editor for IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY and IEEE COMMUNICATIONS LETTERS. He is also an Area Editor of IEEE TRANSACTIONS ON MACHINE LEARNING IN COMMUNICATIONS AND NETWORKING.