Bilecik Şeyh Edebali Üniversitesi Fen Bilimleri Dergisi
Bilecik Seyh Edebali University Journal of Science
11(2), 369-384, 2024
DOİ: 10.35193/bseufbd.1381786
e-ISSN: 2458-7575 (https://dergipark.org.tr/tr/pub/bseufbd)

Araştırma Makalesi - Research Article

Interactive Use of Deep Learning and Ethereum Blockchain for the Security of IIoT Sensor Data

IIoT Sensör Verilerinin Güvenliği için Derin Öğrenme ve Ethereum Blok Zinciri'nin İnteraktif Kullanımı

Emrullah Şahin[1*], Naciye Nur Arslan[2], Fırat Aydemir[3]

Geliş / Received: 27/10/2023 ***Revize / Revised: 30/12/2023*** ***Kabul / Accepted: 03/01/2024***

ABSTRACT

The Industrial Internet of Things (IIoT) refers to a structure where multiple devices and sensors communicate with each other over a network. As the number of internet-connected devices increases, so does the number of attacks on these devices. Therefore, it has become important to secure the data and prevent potential threats to the data in factories or workplaces. In this study, a deep learning-based architecture was used to determine whether the data collected from IIoT sensors was under attack by looking at network traffic. The data that was not exposed to attacks was stored on the Ethereum Blockchain network. The Ethereum blockchain network ensured that sensor data was stored securely without relying on any central authority and prevented data loss in case of any attack. Thanks to the communication process over the blockchain network, updating and sharing data was facilitated. The proposed deep learning-based intrusion detection system separated normal and anomaly data with 100% accuracy. The anomaly data were identified with an average of 95% accuracy for which attack type they belonged to. The data that was not exposed to attacks was processed on the blockchain network, and an alert system was implemented for the detected attack data. This study presents a method that companies can use to secure IIoT sensor data.

Keywords- Industrial Internet of Things, Blockchain, Cyber Security, Intrusion Detection System, Deep Learning

ÖZ

Endüstriyel Nesnelerin İnterneti (IIoT), birden fazla cihazın ve sensörün bir ağ üzerinden birbirleriyle iletişim kurduğu bir yapıyı ifade eder. İnternetle bağlantılı cihazların sayısı arttıkça, bu cihazlara yönelik saldırıların sayısı da artar. Bu nedenle, fabrikalarda veya işyerlerinde verileri güvence altına almak ve olası tehditlere karşı önlem almak önemli hale gelmiştir. Bu çalışmada, IIoT sensörlerinden toplanan verilerin ağ trafiğine bakılarak saldırı altında olup olmadığını belirlemek için derin öğrenme tabanlı bir mimari kullanıldı. Saldırıya uğramamış veriler Ethereum Blok Zincir ağına kaydedildi. Ethereum blok zincir ağı, sensör verilerinin merkezi bir otoriteye dayanmadan güvenli bir şekilde saklanmasını ve herhangi bir saldırı durumunda veri kaybının önlenmesini sağlamaktadır. Blok zincir ağı üzerinden iletişim süreci sayesinde veri güncelleme ve paylaşımı kolaylaştırıldı. Önerilen derin öğrenme tabanlı saldırı tespit sistemi, normal ve anormal verileri %100 doğrulukla

[1]*Corresponding author contact: emrullah.sahin@dpu.edu.tr (https://orcid.org/0000-0002-3390-6285)
Software Engineering, Faculty of Engineering, Kutahya Dumlupinar University, Kutahya Turkey
[2]Contact: naciye.arslan@dpu.edu.tr (https://orcid.org/0000-0002-3208-7986)
Software Engineering, Faculty of Engineering, Kutahya Dumlupinar University, Kutahya Turkey
[3]Contact: firat.aydemir@dpu.edu.tr (https://orcid.org/0000-0002-8965-1429)
Computer Engineering, Faculty of Engineering, Kutahya Dumlupinar University, Kutahya Turkey

ayırabilmektedir. Anormal verilerinde, hangi saldırı tipine ait oldukları ortalama %95 doğrulukla tanımlandı. Saldırılara maruz kalmayan veriler blok zincir ağında işlendi ve tespit edilen saldırı verileri için uyarı sistemi geliştirildi. Bu çalışma, şirketlerin IIoT sensör verilerinin güvenliğini sağlamak için kullanabileceği bir yöntem sunmaktadır.

Anahtar Kelimeler- Endüstriyel Nesnelerin İnterneti, Blok Zinciri, Siber Güvenlik, Sızma Tespit Sistemi, Derin Öğrenme

I. INTRODUCTION

The Internet of Things (IoT) is a network that connects real-life objects such as devices, vehicles, and appliances. These objects are equipped with sensors, software, and connectivity features, allowing them to gather and share data over the Internet. IoT is utilized in various fields such as Industry 4.0, smart cities, healthcare, and transportation [1]. Since IoT devices are connected to the internet, systems are vulnerable to attacks [2]. With the increasing number of IoT devices, the demand for cybersecurity solutions has also risen. The International Data Corporation (IDC) predicts that by 2025, there will be 55.7 billion connected devices, and generated data by these devices will significantly increase from 18.3 zettabytes in 2019 to 73.1 zettabytes [3]. Therefore, it is expected that the cybersecurity market will reach $270 billion by 2026.

The Industrial Internet of Things (IIoT) is a subset of IoT and consists of a collection of interconnected devices, sensors, networks, and software that encompass all stages of a manufacturing process or industry. IIoT systems are utilized to enhance the performance, efficiency, and security of industrial processes or machine operations [4]. However, when system security is compromised, processes can fail, or systems may become unable to fulfil their functions. These security vulnerabilities can result in production losses, increased costs, or customer attrition. Additionally, they can lead to the unauthorized acquisition or use of personal data belonging to employees or customers by third parties.

IoT intrusion detection is defined as any unauthorized action or activity that causes harm to the IoT ecosystem [5]. It refers to unauthorized access to data or devices, alteration or destruction of data or devices, and disruption of the normal functioning of the IoT system [6]. Examples of IoT intrusion detection include malicious software attacks, denial of service attacks, and unauthorized access to devices. With the increasing number of connected devices, IoT intrusion detection has become a significant concern in the field of cybersecurity. Deep learning is utilized as an effective tool in IoT intrusion detection. The IoT ecosystem typically generates a large amount of data, and analyzing and protecting it against intrusions can be challenging using traditional methods. Deep learning can process these large datasets and identify meaningful patterns and behaviors [7].

IIoT and blockchain technology can be combined to enhance the security of industrial systems and processes [8]. Blockchain is a distributed data structure that records transactions without the need for a central authority. Each block in a blockchain contains multiple transactions, and when a new transaction occurs on the blockchain, the record of that transaction is added to the copies of the ledger held by all participants [9]. Blockchain transactions are secured using cryptography, and the ledger is maintained by multiple computers rather than a central authority. Therefore, making changes to historical transaction records requires modifying each copy, which is highly challenging. The identities of the parties involved in transactions are encoded and protected. As a result, blockchain technology has become a popular choice for applications such as digital currencies, supply chain management, and voting systems [10].

In the context of IIoT, blockchain can be used to create an immutable record of sensor data and system events, which can be utilized to detect and prevent security threats [11]. For example, data collected by an IoT device can be recorded on the blockchain network and accessed by other devices, ensuring data integrity and preventing data manipulation. Furthermore, transactions conducted through blockchain technology are performed securely and encrypted, thereby enhancing the security of IoT devices [12]. In the tracking process of products manufactured in a factory, data collected by IoT sensors can be recorded on the blockchain to enable tracking information such as the current stage of products and the devices involved in processing [13]. Another example is the combined use of blockchain and IoT in the energy market, where IoT devices present in many homes can monitor energy production and consumption in real-time. This data can be recorded on the blockchain to facilitate the creation of a fair energy market between energy producers and consumers. Additionally, energy transactions conducted through blockchain technology are executed securely and transparently.

Blockchain technology, unlike traditional databases, enables trust between parties involved in transactions without the need for a central authority. Transactions on the blockchain network are shared fairly among the participating parties, and no single entity can control the network. This demonstrates that blockchain can be used as a secure and reliable means for financial transactions, industrial data storage, and commercial operations [14].

When delving into the fundamentals of why blockchain is considered trustworthy, we encounter the consortium mechanism. A blockchain consortium is a group of organizations that come together to operate a blockchain network [15]. Blockchain consortiums typically use Proof-of-Stake (PoS) variations as a consensus algorithm, maintaining blockchain integrity and verifying transactions while ensuring security and stability [16]. However, depending on trust levels within the consortium, other methods such as Proof of Authority (PoA) and Practical Byzantine Fault Tolerance (PBFT) may be preferred. For a democratic approach, Delegated Proof of Stake (DPoS) allows token holders to elect validators. For specific needs, consortiums can integrate various consensus algorithms using a pluggable consensus method. These range from Federated Byzantine Agreement (FBA), Proof of Elapsed Time (PoET) to traditional methods like Proof of Work (PoW) and PoS [17,18]. As technology advances, new consensus methods will emerge to meet evolving requirements.

In our study, we proposed a deep learning-based intrusion detection system integrated with blockchain. To prevent attacks on industrial IoT devices, we first check if the incoming data to our system has been compromised. If there is malicious data as a result of an attack, we determine the type of attack it belongs to. We store IoT sensor data containing normal network traffic that is not exposed to attacks in the blockchain system.

The following sections will delve into a deeper analysis and comparisons with other studies in the field. Investigations related to IoT threats and anomaly detection from various studies will be examined in Section I. Detailed insights into the definition of the dataset, a range of attacks and anomalies, learning models, and system frameworks are provided in Section II. Section III encompasses our experiment methodology, findings from the analysis, and comparisons with the most recent and advanced techniques in the field. Lastly, Section IV, and V, discloses the conclusions of our study, along with a perspective on potential future applications.

A. Related Works

A review of studies on the security of IoT devices in the literature reveals three distinct phases: machine learning, deep learning, and blockchain-related research.

The authors propose an Intrusion Detection System (IDS) for detecting injection attacks in smart cities. They employ two feature selection techniques, namely constant removal and recursive feature elimination, and test them with Decision Tree (DT), Random Forests (RF), and Support Vector Machines (SVM) classifiers. They use the AWID dataset, which contains real Wi-Fi traces. By utilizing only 8 selected features through constant removal and recursive feature elimination, they achieve the highest accuracy rate (99%) with the DT classifier [3].

In their study, the authors created a simulated dataset for botnet attacks. They compare SVM, LSTM, and RNN models to classify the attacks using the generated dataset. They present binary classification results for both attacked and non-attacked attack types. Using SVM with all features, they achieve the highest accuracy rate [1].

Using a specifically designed IoT/IIoT testbed that includes a range of typical devices, sensors, protocols, and cloud/edge setups, Ferrag et al. produce a dataset. Data from various IoT devices, including inexpensive digital sensors, ultrasonic sensors, water level detection sensors, pH sensor probes, soil moisture sensors, heart rate sensors, flame sensors, and others are included in the dataset. Along with developing the dataset, they also identify and analyze fourteen attacks that they group into five categories: information gathering, man-in-the-middle attacks, injection attacks, and other attacks related to IoT and IIoT communication protocols. To identify these attacks, they contrast deep learning with conventional machine learning methods (DT, RF, KNN, and SVM) [19].

Babu et al. aim to address security concerns in smart cities. They propose a permission-based blockchain system using the mediator PUF (Physically Unclonable Function) model. Additionally, they propose a collaborative detection system utilizing machine learning techniques (SVM, RF, LR, DT) to detect DDoS attacks on IoT devices. Their ensemble model (SVM+RF+LR+DT) achieves the highest accuracy rate (97.39%) [20].

Tahir et al. propose a new authentication and authorization framework for blockchain-enabled IoT networks in healthcare applications. The framework utilizes a probabilistic model that uses random numbers in the authentication process and establishes a secure connection between IoT devices for data collection. The framework addresses critical security, privacy, and legal requirements in healthcare informatics. The proposed model is evaluated using the AVISPA tool and the Cooja simulator [21].

Chen et al. propose a blockchain-based system for Healthcare IoT that addresses isolated information and security gaps. The proposed system focuses on balancing data sharing and privacy protection. To achieve this, they designed a privacy protection method based on content extraction signatures, which provides detailed privacy protection to patients. Additionally, they designed a fault-tolerant Byzantine leader selection mechanism to enhance the security of the Raft algorithm while maintaining data sharing efficiency. Lastly, they design a summary contract to ensure efficient data acquisition. The proposed mechanism is evaluated through simulation and analysis for efficiency and security [22].

When reviewing the literature, it is observed that most studies perform intrusion detection on network traffic datasets obtained from IoT sensors or attempt to enhance security by storing IoT data in a blockchain mechanism. In our proposed system, we have a two-phase security mechanism. In the first phase, intrusion detection is performed, and if a sensor is under attack, our alert system is activated to provide information about the type of attack. If data is received from an unaffected sensor, the second phase of our work, which is the blockchain mechanism, records the sensor data. A similar architecture to our proposed system has not been found in the literature.

II. MATERIAL AND METHODS

A. Dataset

In this study, the EdgeIIotset dataset prepared specifically for deep learning algorithms was used to check for any intrusion or tampering in the data received from the sensors. The EdgeIIotset dataset is a new cybersecurity dataset that can be used for IoT and IIoT applications [19]. Researchers have created this dataset to enable machine learning algorithms to perform intrusion and tampering detection on sensor data. Additionally, the researchers of the IIoT-Edge dataset prepared a subset called DNNEdgeIIoT-dataset to test deep learning-based IDS. IoT and IIoT detection, Cloud computing, Blockchain network, Fog computing, NFV, SDN, and Edge computing are the seven layers that make up the dataset. It contains network traffic information gathered from a variety of sensors, including ultrasonic, water level detection, pH, soil moisture, pulse rate, and flame sensors. The report also includes information on 14 distinct attacks against IoT and IIoT connectivity protocols. Denial-of-Service/Distributed denial-of-service, Information gathering, Man-in-the-middle, Injection, and Malware are the five basic categories into which these types of attacks fall.

In our data preprocessing phase, after removing rows with missing values and splitting the dataset into training (80%) and testing (20%) sets, we focused on converting categorical output values into numerical form. For this conversion, we utilized pandas' dummy encoding method. This technique transforms each categorical feature into multiple binary columns, representing the presence (1) or absence (0) of each unique value in the original feature.

Mathematically, for a categorical variable C with m unique values, dummy encoding creates m new binary features. Each feature D_i corresponds to one of the m values, where $D_i = 1$ if C equals the $i^{\{th\}}$ value, and $D_i = 0$ otherwise. This method ensures that the model interprets these features as distinct categories without any inherent order.

This process, along with the normalization of data to a range between 0 and 1, prepared our dataset for network processing. The transformed class distribution and statistical values are presented in Figure 1, reflecting the dataset's state after these preprocessing steps.

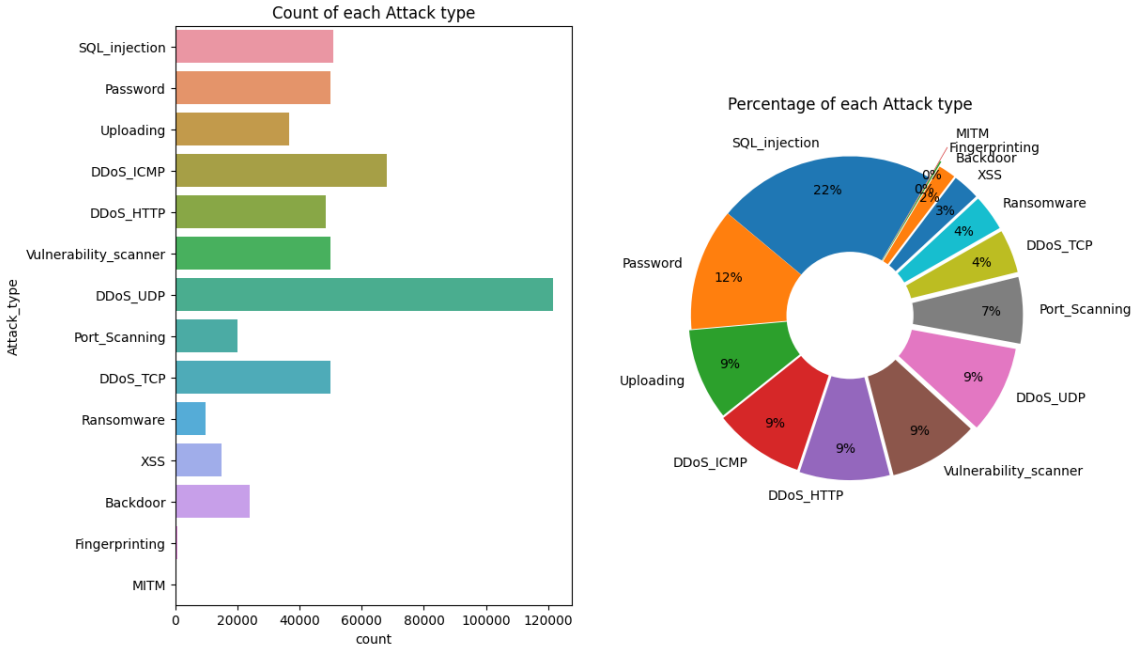

Figure 1. Distribution graphs of the EdgeIIoT dataset based on classes.

B. *Feature Selection and SmoteENN*

In our study, to determine the best features in the dataset and thereby achieve improved results, the Recursive Feature Elimination with Cross-Validation (RFECV) technique [23,24] is employed. This method systematically combines feature elimination with model validation, executed through the following steps:

Initial Setup:

- Let $F = \{f_1, f_2, \ldots, f_{n\}}$ represent the set of all n features in the dataset.
- A Random Forest classifier, denoted as E, is utilized as the base estimator. This choice is motivated by the classifier's capability to compute feature importances, which are essential for the elimination process.

Recursive Feature Elimination (RFE):

- The importance of each feature is evaluated by the estimator E. In the case of the Random Forest classifier, this is typically based on metrics like Gini importance or mean decrease in impurity.
- The feature with the least importance, say f_{least}, is identified and eliminated, resulting in a reduced feature set $F' = F \setminus \{f\_least\}$.

Cross-Validation (CV) Integration:

- The dataset is partitioned into k equal-sized subsets for k-fold cross-validation.
- For each iteration, the estimator E is trained on $k - 1$ subsets and validated on the remaining subset.
- The performance of E is evaluated, typically using metrics like accuracy or F1-score, on the validation subset.
- This process is repeated until each subset has been used for validation once.

Optimization and Selection:

- The average cross-validation score is calculated for each reduced feature set F'.
- The process iteratively continues, eliminating one feature at a time, and tracking the corresponding cross-validation scores.
- The optimal number of features is determined when the highest average cross-validation score is achieved.

Outcome:

- The final set of selected features, denoted as F^*, represents the subset of F that yields the highest cross-validation score, balancing model performance and simplicity.

This methodical approach, underpinned by the rigorous application of RFECV, led to the identification of the most relevant features in our dataset, significantly contributing to the enhanced performance of our model. As a result of this process, from the initial set of 92 features, we successfully identified and selected the 27 most impactful features. These selected features, which represent the optimal subset for our model, are comprehensively listed in Table 1. This strategic reduction and focus on key features have been instrumental in improving the model's accuracy and efficiency.

Table 1. List of the best 27 features obtained using the RFECV method.

No	Feature Name	No	Feature Name	No	Feature Name
1	icmp.checksum	10	tcp.connection.syn	19	http.request.method-0.0
2	icmp.seq_le	11	tcp.connection.synack	20	http.referer-0.0
3	http.content_length	12	tcp.flags	21	http.request.method-GET
4	http.response	13	tcp.flags.ack	22	http.referer-0
5	tcp.ack	14	tcp.len	23	http.referer-127.0.0.1
6	tcp.ack_raw	15	tcp.seq	24	http.request.version-0
7	tcp.checksum	16	udp.stream	25	http.request.version-0.0
8	tcp.connection.fin	17	dns.qry.name	26	http.request.version-HTTP/1.0
9	tcp.connection.rst	18	http.request.method-0	27	http.request.version-HTTP/1.1

To address the class imbalance problem in the dataset, the SMOTEENN (Synthetic Minority Over-sampling Technique with Edited Nearest Neighbors) technique was applied. Class imbalance occurs when one class has significantly more instances than the others. In such cases, machine learning models tend to focus more on the majority class and may misclassify minority class examples. SMOTE is a popular high-speed oversampling technique that generates synthetic examples for the minority class by interpolating among existing examples [25]. However, SMOTE can also introduce noisy examples that may degrade the classifier's performance. On the other hand, ENN is an undersampling technique that removes examples from the majority class that is close to the minority class in the feature space [26]. Since ENN can help in removing the noisy examples generated by SMOTE, it is deemed appropriate to use both methods together. After these procedures, the proportional distribution of the resulting data across the classes is shown in Figure 2.

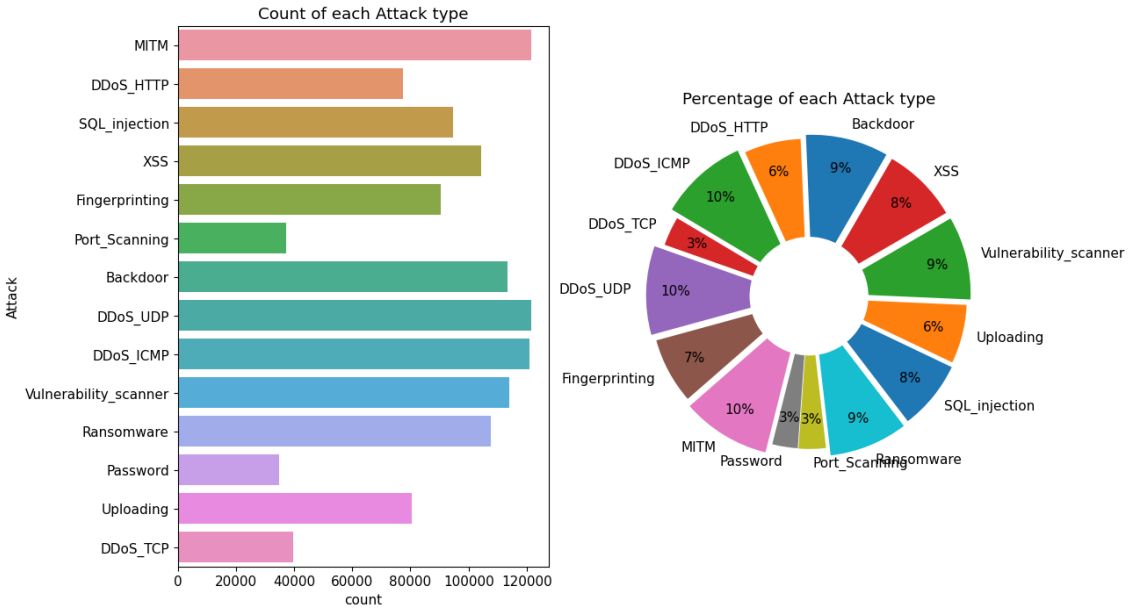

Figure 2. Distribution of data after applying the SmoteENN method for data augmentation.

C. Deep Learning based Attack Detector

After preprocessing the data, the Anomaly Detector model is applied. The Anomaly Detector model performs binary classification, separating the data into two classes: attacked (anomaly) and not attacked (normal). If no anomalies are detected in the incoming sensor data, the data is stored in the Blockchain layer. If an attack is detected in the incoming data, the Intrusion Detector model is used to determine the type of attack among the 14 predefined attack types. The Warning layer is activated, providing the necessary information to the expert using the system. The proposed architectural structure is presented in Figure 3.

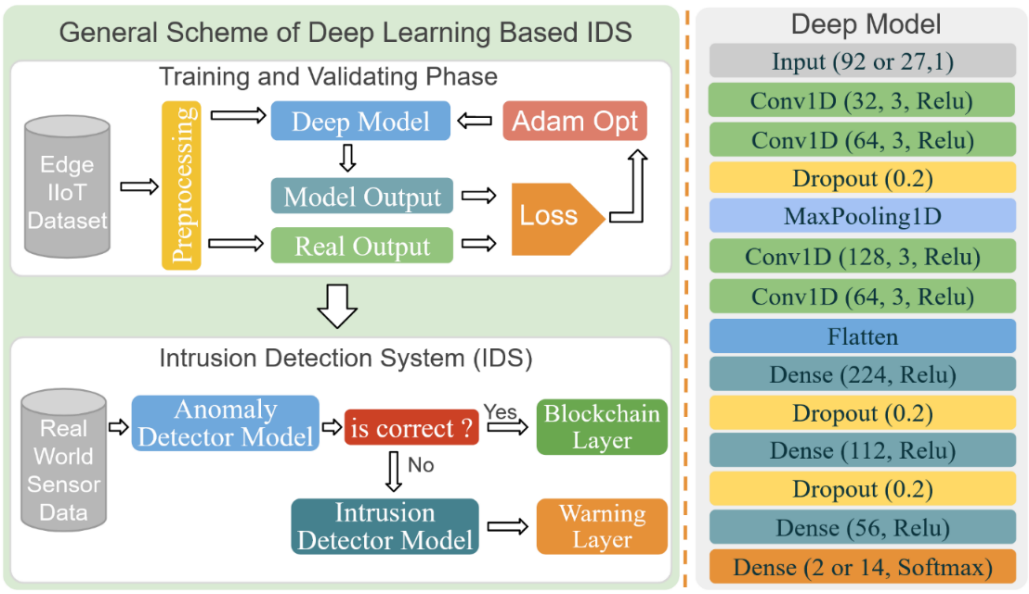

Figure 3. Main diagram of the proposed Deep Learning-based Intrusion Detection System.

The proposed Anomaly and Intrusion Detector models share the same layer composition and workflow. The recommended model starts with two convolutional layers with filter sizes of 32 and 64, and kernel size of 3. A Dropout layer with a rate of 0.2 is added to reduce overfitting by randomly disabling the outputs of neurons [27]. Then, a Max pooling layer is added to reduce the input size by selecting the maximum feature values. Two more convolutional layers with filter sizes of 128 and 64, and kernel size of 3 are added. All convolutional layers use the ReLU activation function to calculate their outputs. A Flatten layer is added to convert the inputs into a vector form [28]. A fully connected layer with 224 neurons and a ReLU activation function is added. Another Dropout layer with a rate of 0.2 is added. A fully connected layer with 112 neurons using the ReLU activation function is added. Another Dropout layer is added. A fully connected layer with 56 neurons is added. The last layer of the model includes the SoftMax method.

D. Blockchain-based IIoT Data Management

Blockchain technology is a method used to enhance the security of digital transactions. This technology enables the establishment of direct trust between the parties involved in the transactions and guarantees the irreversibility of the transactions. The blockchain network is distributed fairly among the parties involved in the transactions, and it is not possible for any party to control the network.

Currently, there are multiple blockchain technologies available. In this study, the Ethereum blockchain network is used. The Ethereum blockchain network is a distributed computer network and an open-source platform that enables the execution of smart contracts [29]. This network is operated by numerous computers and nodes worldwide, and each contributes to the security of the network and the accuracy of the transactions. As a result, the Ethereum network enables secure and fast transactions without the need for a central authority [30]. The Ethereum network verifies transactions using cryptographic techniques and maintains data integrity among all participating nodes. This ensures that all transactions performed on the Ethereum network are carried out securely and transparently.

The Ethereum blockchain network proffers a multitude of advantages primarily due to its prowess in executing smart contracts [31]. It operates on the principle of decentralization, nullifying the need for a central authority, thereby facilitating expeditious and secure transactions. Additionally, Ethereum ensures rigorous security through its verification of transactions using advanced cryptographic techniques, thereby maintaining data integrity across all nodes. Consequently, transactions on the network are secure and resistant to tampering. The speed at which transactions are verified and confirmed is substantially accelerated due to Ethereum's distributed architecture. The hallmark of Ethereum, smart contracts, automate a wide array of processes, thereby mitigating human errors. Moreover, Ethereum offers a transparent transaction framework where all transactions can be tracked and recorded by any observer, boosting transaction transparency and significantly reducing the likelihood of manipulation.

Thanks to these advantages, the Ethereum blockchain network is used in various sectors. In our study, we utilized the Smart Contract feature of the Ethereum network for the storage and management of data from sensors in factories. These contracts are written in the Solidity programming language. Solidity is the most popular programming language for smart contracts on the Ethereum platform [32]. Solidity has a syntax similar to other languages such as C++ and JavaScript. Smart contracts written in Solidity are distributed and executed among the nodes in the Ethereum network.

Once the contract code is written, it is compiled into executable bytecode by the Ethereum Virtual Machine (EVM). The contract code is then sent to the Ethereum network, where it is distributed among the nodes of the network. When a user submits a transaction that affects the contract, the transaction is verified on the network, and the transaction fee required for executing the contract code is paid in Ethereum's cryptocurrency, Ether. Once the transaction is verified and the fee is paid, the contract code is executed by the nodes in the network, and the results are returned to the network.

In this way, you can write smart contracts using Solidity on the Ethereum blockchain network, send them to the network, and perform various transactions through these contracts. Another advantage of using the Ethereum network is its compatibility with IPFS data servers for easy data transfer. IPFS (InterPlanetary File System) is a distributed file system and protocol [31]. The integration of Ethereum with IPFS allows Ethereum smart contracts to hold references to files stored on IPFS and access those files. This integration enables smart contracts on Ethereum to be connected to a distributed file system like IPFS instead of relying on centralized servers.

Technically, the integration between IPFS and Ethereum provides Ethereum smart contracts with the ability to access files stored on IPFS. This is achieved by making IPFS hashes (IPFS addresses) part of the Ethereum smart contracts. A user can store the hash of an IPFS file within an Ethereum smart contract. As a result, the file content is not uploaded to the Ethereum network but is stored on IPFS. IPFS hashes can be placed in the data fields or function parameters of Ethereum smart contracts and later used to access the file.

In summary, in this study, a secure system was developed for the storage, verification, processing, and management of data from IoT sensors in factories. The visual diagram of the proposed Blockchain-based Data Management System is presented in Figure 4. The step-by-step progression of this system is as follows:

1. *Blockchain registration layer*: Allows factories to register with their company information and then register their sensors.
2. *Data reading layer*: Involves transferring data from IoT sensors to distributed edge devices.
3. *Data validation layer*: Involves testing the data transferred to edge devices with a pre-trained Deep Learning-based Intrusion and Intrusion Detection System to check for tampering or attack attempts. If any tampering or attack is detected, send a message to the alert layer in the fifth step; if not, transfer the data to the blockchain layer in the fourth step.
4. *Blockchain data addition layer*: The factory's user and sensor information are verified through the Ethereum network. If the transaction is approved, the validated data from IoT sensors is stored on the IPFS server. During the registration process, the access key obtained from the IPFS server is transferred to the Ethereum network along with the factory and sensor information.
5. *Alert message sending*: The sensor and type of attack information from the alert message are transferred to the factory's alert system.

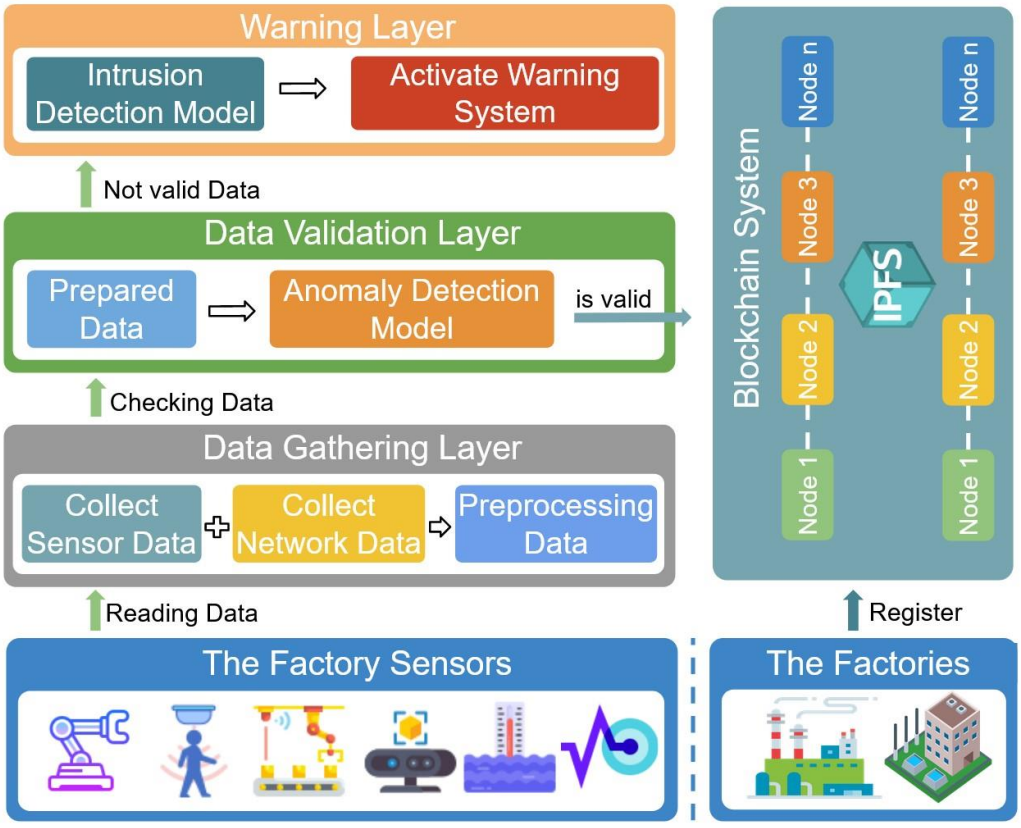

Figure 4. Main diagram of the proposed Blockchain-based Data Management System.

III. RESULTS

A. *Blockchain Implementation*

This study was developed using the Ethereum blockchain. The main coding part was done in the online Remix environment provided by the Ethereum blockchain. This environment allows us to write Smart Contracts and test them on various blockchain networks. Our study works on a private blockchain that we created. The first of our Smart Contracts is the Factory Registration contract. The pseudocode of this contract is provided in Figure 5. The main function of this algorithm is factory Register, which takes factory_name, registry_number, and tax_number as parameters. In this function, the readBlockchain function is called to retrieve the data from the chain and assign it to the "blockchain" variable. Then, if there is a previous record for this factory, it displays a warning message. If there is no previous record, it combines the incoming three parameters with the timestamp value and passes it to the writeBlockchain function.

```
Function factoryRegister(factory_name, registry_number, tax_number)
    blockchain ← call readBlockchain()
    if registry_number in blockchain then
        print "This registry number is already in use."
    else
        data ← [factory_name, tax_number, registry_number, timestamp]
        call writeBlockchain(data)
    endif
endFunction
```

Figure 5. Factory registration function.

Our second Smart Contract in our application is the Sensor Registration. After the factory registration phase, the sensors belonging to these factories are registered on the blockchain along with their MAC addresses. The pseudocode of this contract is provided in Figure 6. First, similar to the first algorithm, data is read from the blockchain. Then, it is determined whether the sensor has been previously registered based on the MAC address. If there is no existing record and the incoming factory_id (factory information) is also validated, the sensor information along with its corresponding factory_id is passed to the writeBlockchain function for registration on the blockchain network.

```
Function sensorRegister (sensor_name, mac_address, factory_id)
    blockchain ← call readBlockchain()
    if mac_address in blockchain then
        print "This Sensor MAC address is already in use."
    else if factory_id not in blockchain
        print "The factory with the given ID does not exist."
    else
        data ← [sensor_name, mac_address, factory_id, timestamp]
        call writeBlockchain(data)
    endif
endFunction
```

Figure 6. Sensor registration function.

The data from the sensors go through the data Control function before being sent to the blockchain network. The pseudocode for this function is provided in Figure 7. We pass the data from the sensors and the network's data to our pre-trained Anomaly Detector model to perform anomaly detection. If an anomaly or attack is detected, we pass the data to our Intrusion Detector model, which performs intrusion detection and returns the type of attack. The information is then directed to the warning Layer. If the data is determined to be healthy, we send the sensor data to the sensorAddData function for further processing.

```
Function dataControl(sensor_id, sensor_data, network_data)
    anomaly_result ← call AnomalyDetectorModel([sensor_data, network_data])
    if anomaly_result then
        intrusion_result ← call IntrusionDetectorModel([sensor_data, network_data])
        call warningLayer(ıntrusion_result)
    else
        call sensorAddData(sensor_id, sensor_data)
endFunction
```

Figure 7. Data control function.

The verified data obtained in the final stage is transferred to the sensor Add Data function. The pseudocode for this function is provided in Figure 8. In the initial step, data is retrieved from the blockchain network, and the sensor information is checked. If the sensor does not exist, a warning is issued. If the sensor information is validated, the incoming data is uploaded to the IPFS (Inter Planetary File System) using the upload IPFS function. The resulting IPFS hash value is then sent to the blockchain network for data access.

```
Function sensorAddData(sensor_id, sensor_data)
    blockchain ← call readBlockchain()
    if sensor_id not in blockchain then
        print "Sensor ID not found in the blockchain."
    else
        ipfs_hash ← call uploadIPFS(sensor_data)
        data ← [sensor_id, ipfs_hash, timestamp]
        call writeBlockchain(data)
endFunction
```

Figure 8. Sensor data recording function.

1) Gas Cost Analysis and Considerations: In this analysis, we focus on the gas costs associated with the "Factory Registration", "Sensor Registration", and "Sensor Data Manager" smart contracts on the Ethereum network. The values are presented in Wei, the smallest denomination of Ether, and represent average costs obtained in the Remix VM (Shanghai) environment. The cost calculations are provided in Table 2.

Table 2. Gas Costs of Smart Contracts (in Wei).

Smart Contract	Operation	Transaction Cost (Wei)	Execution Cost (Wei)
Factory *Registration*	Create	140,000	120,000
	Update	40,000	20,000
	Delete	50,000	40,000
Sensor Registration	Create	160,000	140,000
	Update	50,000	30,000
	Delete	50,000	40,000
Sensor Data Manager	Create	170,000	55,000
	Delete	55,000	45,000

*Considerations on IPFS Costs and Real-world Testing***:** It's important to note that the above gas cost analysis does not include the costs associated with IPFS (InterPlanetary File System). Factories can either host IPFS software on their servers or use third-party hosting services, leading to variability in IPFS-related expenses. Additionally, while the simulations provide useful insights, testing this blockchain application in a real-world factory setting would yield more definitive results.

Scenario Analysis: Gas Cost for 200 Sensors: Let's consider a hypothetical scenario where a factory has 200 sensors, with each sensor transmitting data every five minutes. Assuming the `Create` operation is used for each data transmission in the `Sensor Data Manager` contract, we can estimate the total gas cost over a certain period. For simplicity, we'll calculate the cost for one hour:

- Number of data transmissions per sensor in one hour: $\frac{60\ minutes}{5\ minutes\ /\ sensor}$ = 12 transmissions/sensor.
- Total transmissions for 200 sensors in one hour: 12 transmissions / sensor x 200 sensors = 2400 transmissions.
- Gas cost per transmission (using `Create` in Sensor Data Manager): 170,000 Wei.
- Total gas cost for one hour: 2400 transmissions x 170,000 Wei/transmission.

Now, let's calculate the total gas cost for one hour: Total Gas Cost for 1 Hour = 2400 x 170,000 Wei.

This calculation provides a concrete example of the operational costs associated with this system for a specific use case.

B. Comparison Metrics

Precision, recall, F1-score, and accuracy are metrics used to evaluate the performance of classification models in fields such as machine learning, deep learning, and data science. These metrics are computed according to the following formulas (1-4):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F1-score} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}} \tag{3}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{4}$$

Here, *TP*, *TN*, *FP*, *FN*, and *FP* indicate the number of true positive predictions, true negative predictions, true false positive predictions, and true negative predictions, respectively. These indicators have a range of 0 to 1, with 0 being the worst performance and 1 the best performance. While accuracy can be more helpful in datasets with balanced class distributions, precision, recall, and F1-score are more insightful in datasets with class imbalance.

Confusion Matrix is a matrix used to evaluate the performance of classification models. The matrix displays the numerical values of true and predicted classes, distinguishing correct classifications,

misclassifications, false positives, and false negatives. Simply put, the Confusion Matrix is a visual tool that depicts the classification ability of a model.

C. Loss Functions

In training our anomaly detection model, we focused on distinguishing between normal and anomalous instances. This task, being a binary classification problem, necessitated the use of the Binary Cross-Entropy (BCE) loss function. The BCE [33] loss is calculated as $-(y\log(p) + (1-y)\log(1-p))$, where y is the true label (0 or 1 for normal and anomalous instances, respectively), and p is the predicted probability of the instance being anomalous. This function is particularly effective in measuring the difference between the predicted probabilities and the actual binary labels, penalizing any significant deviations.

For the intrusion detection model, which involves classifying multiple types of network intrusions, the Categorical Cross-Entropy (CCE [34]) loss function was employed due to its suitability for multi-class classification tasks. The CCE loss is defined as $-\sum_{c=1}^{M} y_{o,c}\log(p_{o,c})$ where M represents the number of classes, $y_{o,c}$ is a binary indicator for whether class c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability of observation o being in class c. The CCE loss function excels in scenarios where each observation is assigned to one, and only one of several categories, making it highly suitable for our model focused on intrusion detection.

Both loss functions were integral in guiding the respective models toward achieving accurate and reliable classifications, a critical aspect for the effective detection of anomalies and intrusions in network environments.

D. Anomaly Detection Results

Our Anomaly Detector model, developed in the Keras framework to discern attacks on sensor data, underwent a rigorous training regimen. It was trained using the Adam optimizer over 30 epochs with a batch size of 256. The training dataset, constituting 80% of the total data, encompassed ~1,500,000 samples, while the test dataset, making up the remaining 20%, contained ~400,000 samples. The performance metrics of the model on this test dataset are illustrated in Table 3.

Table 3. Performance metrics of the Anomaly Detector model on the test dataset.

Dataset Version	Class Name	Precision	Recall	F1-Score	Accuracy
Standard dataset	Normal	0.89	0.99	0.94	0.91
	Anomaly	0.98	0.68	0.80	
Pre-processed dataset	Normal	1.00	1.00	1.00	1.00
	Anomaly	1.00	1.00	1.00	

Despite the dataset exhibiting a significant class imbalance, with a ~3.5:1 ratio between the two classes, the model achieved notable results on the standard dataset. It showed high precision in detecting normal data, at 0.89, with an overall accuracy of 0.91. The model's precision for anomaly detection was impressive at 0.98, although recall and F1-scores were relatively lower.

The model's advanced capabilities, as discussed in the intrusion detection section, played a pivotal role in these outcomes. It demonstrated a strong aptitude for understanding the data, a critical factor in overcoming the challenges posed by class imbalance. Moreover, when tested on the pre-processed dataset, the model's performance was exceptional, achieving 100% accuracy, precision, and F1-score for both classes. This remarkable performance, even on the test dataset, indicates that the model not only understood the data well but also effectively avoided overfitting, showcasing impressive results across the board.

E. Intrusion Detection Results

We utilize our Intrusion Detector model to determine the specific attack types of the data identified as belonging to the anomaly class by our Anomaly Detector model. This model has the same network structure as the Anomaly Detector, except for the last layer, which contains 14 neurons representing the different attack classes. We trained this model using the Keras framework, performing the training process for 50 epochs with a batch size of 256 and utilizing the Adam optimization method. The training dataset consisted of approximately 60% of the total data, encompassing around 1,000,000 samples. The validation dataset contained approximately 20% of the data, with ~400,000 samples. The remaining 20% comprised the test dataset, also containing ~400,000 samples.

Due to the imbalanced distribution of the 14 attack classes in our dataset, we had to work on multiple versions. The first version includes only the pre-processed results of the standard 92 features in the dataset. The Confusion matrix and ROC area under the curve (AUC) obtained from the test dataset for this version are provided

in Figure 9, and Table 4. When examining these figures, we can observe that the obtained results have achieved high accuracy rates in certain classes but showed very low performance in multiple classes.

Table 4. Performance of Intrusion Detector model on the test dataset for the standard 92-feature model.

Class Name	Precision	Recall	F1-Score
Backdoor	0.96	0.98	0.97
DDoS_HTTP	0.75	0.95	0.84
DDoS_ICMP	1.00	1.00	1.00
DDoS_TCP	0.82	1.00	0.90
DDoS_UDP	1.00	1.00	1.00
Fingerprinting	0.92	0.69	0.79
MITM	1.00	1.00	1.00
Password	0.45	0.89	0.60
Port_Scanning	1.00	0.49	0.66
Ransomware	1.00	0.87	0.93
SQL_injection	0.81	0.21	0.33
Uploading	0.68	0.48	0.56
Vulnerability_scanner	1.00	0.85	0.92
XSS	0.58	0.38	0.46
Accuracy			0.82
Macro avg	0.85	0.77	0.78
Weighted avg	0.86	0.82	0.81

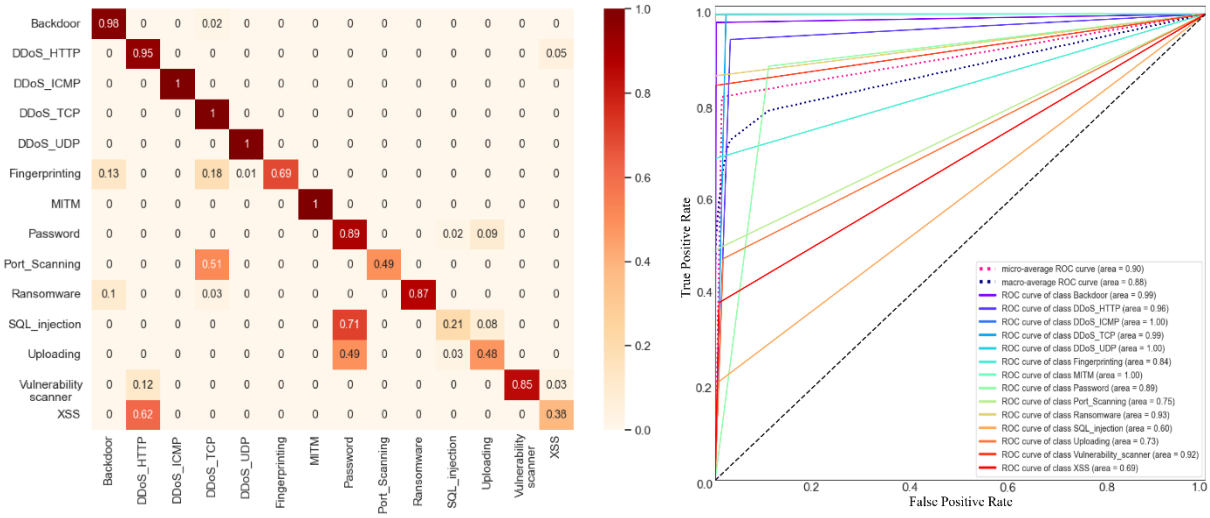

Figure 9. Confusion matrix values (column a) and ROC area indicator (column b) for the standard 92-feature model.

In the initial stage of our multi-class (14 classes) intrusion detection system, we faced a perplexing challenge. Some classes were identified with high accuracy, while others were markedly less accurate. This inconsistency prompted an investigation to ascertain whether the issue was rooted in the model design or the dataset. To this end, we conducted extensive tests on the Resnet50 [35] and Transformer [36] architectures.

The test results were revealing:

- *Transformer*: This model achieved an accuracy of 0.79, an F1-score of 0.74, a recall of 0.78, and a precision of 0.75. Despite these figures, the performance across different classes was uneven.
- *Resnet50*: It showed a better accuracy of 0.93, but like the Transformer, the F1-score was 0.72, recall 0.76, and precision 0.73, again with variable performance across classes.

Ultimately, these tests led us to conclude that the problem did not lie within the network architectures. Rather, it was the dataset itself that was the source of the issue, specifically the class imbalance within it.

To address the imbalance in our dataset, we implemented a multi-faceted strategy focusing on dataset enhancement. In terms of feature selection, we adhered to the methodology outlined in the Material Methods

section B of our study, utilizing the RFECV (Recursive Feature Elimination with Cross-Validation) technique. This approach enabled us to efficiently narrow down from 92 features to the 27 most impactful ones, ensuring a more focused and relevant dataset for our models.

For data augmentation, we primarily utilized the SmoteENN method. While we experimented with various Smote techniques, SmoteENN emerged as the most effective in our context. Its combination of Synthetic Minority Over-sampling Technique (SMOTE) and Edited Nearest Neighbors (ENN) provided a robust solution to our class imbalance issue.

After extensive testing with different combinations of features and data augmentation methods, we found that the best results were achieved with a specific configuration. The combination of using just the 27 selected features, applying class weighting, and incorporating the SmoteENN method for data augmentation proved to be the most successful. This particular setup not only addressed the class imbalance effectively but also enhanced the overall accuracy and reliability of our model, making it adept at handling the complexities of multi-class intrusion detection.

In the final phase of our research, we applied the updated version of our dataset to train our proposed model. The performance of these models is depicted in Figure 10. On the left column of the figure, we present the training and validation accuracy of the models, while the right column features the training and validation loss visual graphs. Careful analysis of these visual graphs shows that our training process was effectively balanced. Both the accuracy and loss visual graphs demonstrate a stable and consistent progression, indicating that our models were adeptly trained, avoiding common pitfalls such as overfitting or underfitting. The steady convergence of these graphs is a testament to the effectiveness of our dataset enhancements, particularly the refined feature selection through RFECV and the implementation of the SmoteENN method for data augmentation. This equilibrium in the training and validation process suggests that our models are not only well-calibrated but also possess a strong capability for generalization, making them robust tools for multi-class intrusion detection.

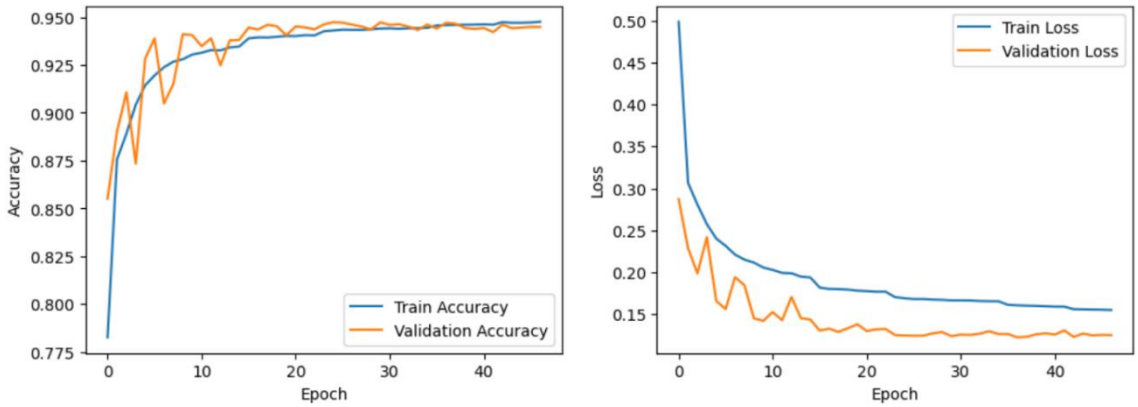

Figure 10. Visual schema of proposed model: Training and validation accuracy and loss graphs.

In our continued effort to thoroughly understand the results of training the proposed model with the updated dataset, we present a detailed analysis in Figure 11. The left column of this figure features the Confusion Matrix, while the right column displays the ROC curves. A comprehensive examination of these elements reveals significant insights. Compared to the initial results shown in Figure 9, the high-performance capability of the network now extends across all classes. This is a clear indication that the modifications we made, particularly in addressing the dataset imbalance, have substantially improved the model's performance. Moreover, a closer look at the individual class performances in the Confusion Matrix and the ROC curves indicates a notable reduction in the imbalance problem, especially in those classes where it was previously more pronounced. This improvement is not just quantitative but qualitative as well, demonstrating the model's enhanced ability to detect attacks accurately and reliably across all classes.

In concluding our results section, we highlight the data presented in Table 5, which outlines the average F1-score, precision, recall, and accuracy for each class in our model. Detailed examination of this table shows a pronounced improvement in comparison to the initial data shown in Table 4. The average accuracy, which was initially in the vicinity of 82%, has impressively risen to approximately 95%. Similarly, the average F1-score, previously at around 81%, has also escalated to about 95%. These enhancements in the average performance metrics across all classes clearly demonstrate the substantial improvements achieved in our model. This leap in average accuracy and F1-scores is a testament to the effectiveness of the adjustments we made to the model and our approach in tackling dataset imbalances. These collective improvements underscore our model's enhanced capability in accurately and reliably detecting multi-class attacks, providing a strong and conclusive finish to our analysis of the results.

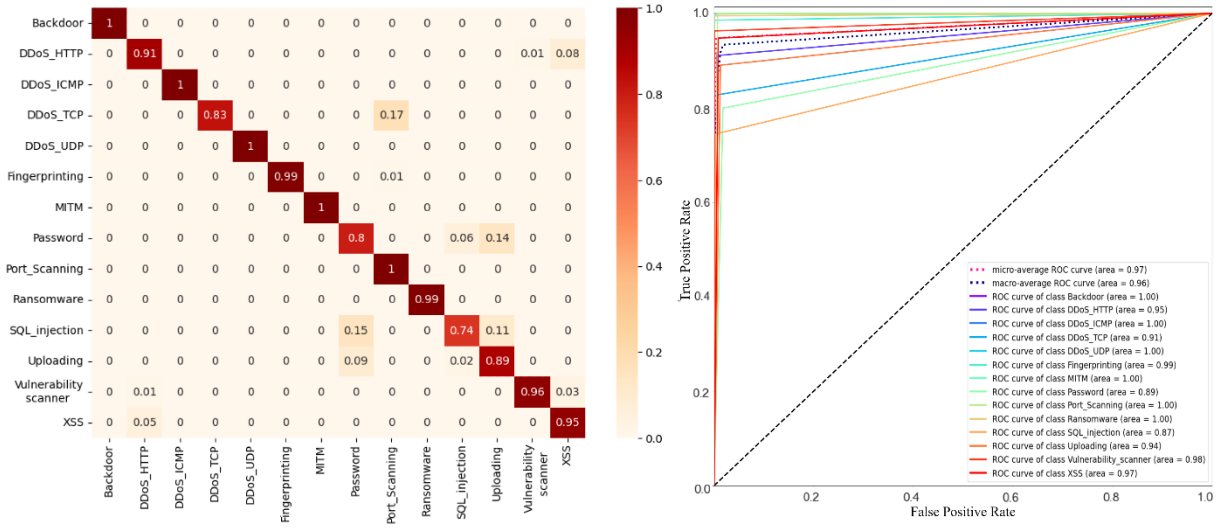

Figure 11. Confusion matrix values (column a) and ROC area indicator (column b) for the improved Intrusion Detector model.

Table 5. Performance of Proposed Intrusion Detector model on the test dataset.

Class Name	Precision	Recall	F1-Score
Backdoor	1.00	1.00	1.00
DDoS_HTTP	0.92	0.91	0.91
DDoS_ICMP	1.00	1.00	1.00
DDoS_TCP	1.00	0.83	0.90
DDoS_UDP	1.00	1.00	1.00
Fingerprinting	0.99	0.99	0.99
MITM	1.00	1.00	1.00
Password	0.56	0.80	0.66
Port_Scanning	0.82	1.00	0.90
Ransomware	1.00	0.99	1.00
SQL_injection	0.95	0.74	0.84
Uploading	0.83	0.89	0.86
Vulnerability_scanner	0.99	0.96	0.98
XSS	0.91	0.95	0.93
Accuracy			0.95
Macro avg	0.93	0.93	0.93
Weighted avg	0.95	0.95	0.95

IV. DISCUSSION

1) Blockchain Implementation Discussion: In this study, we conducted tests on our Ethereum blockchain-based applications named Factory Registration, Sensor Registration, and Sensor Data Manager using the Remix IDE's Virtual Machine environment. These tests included gas calculations, providing us with preliminary insights into the operational aspects of our blockchain implementation. However, it's important to note that the results obtained in a simulated environment, while valuable, may not fully capture the complexities and costs associated with real-world factory settings. Therefore, for more accurate and realistic assessments, especially in terms of cost-effectiveness and practical feasibility, conducting tests in actual factory environments is crucial. This approach will allow us to better understand the scalability, security, and cost implications of deploying our blockchain solution in real-world industrial scenarios, providing a more comprehensive and realistic evaluation.

2) Design of the Proposed Deep Learning-Based IDS: Regarding the design of our proposed deep learning-based Intrusion Detection System, the development process involved experimenting with various layers, filter sizes, kernel dimensions, and activation functions. While the final model emerged as a result of these extensive trials, we acknowledge that the process, though informed by various architectures in the literature, was somewhat prolonged and not fully streamlined. In future work, the adoption of methods like Hyperparameter Optimization (HPO) and Neural Architecture Search (NAS) could significantly enhance the efficiency and effectiveness of the model design process. These advanced techniques would enable a more systematic and optimized exploration of network architectures, potentially leading to more robust and performing models.

Incorporating these methods in future research could streamline the development process, leading to quicker, more efficient iterations and potentially more innovative solutions in the field of intrusion detection.

V. CONCLUSIONS

This study presents a method for ensuring the security of data from IIoT sensors in factories. The security of data is determined based on deep learning techniques to identify whether the data from IIoT sensors have been subjected to attacks. The data that is not exposed to attacks is stored and decentralized on the Ethereum blockchain network, ensuring its security and preventing data loss in the event of an attack. Ethereum blockchain is a distributed database that securely stores data without relying on any central authority. This method can be considered as a significant step towards securing IIoT sensor data. Secure storage of data prevents businesses from experiencing significant data loss. Additionally, the process of decentralization facilitates data updates and sharing. In the future, this method can be further improved to provide a stronger solution for securing IIoT sensor data. This study encourages further research in the field of IIoT security.

REFERENCES

[1] Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, *100*, 779-796.

[2] Hasan, M., Islam, M. M., Zarif, M. I. I., & Hashem, M. M. A. (2019). Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. *Internet of Things*, *7*, 100059.

[3] Gaber, T., El-Ghamry, A., & Hassanien, A. E. (2022). Injection attack detection using machine learning for smart IoT applications. *Physical Communication*, *52*, 101685.

[4] Puri, V., Priyadarshini, I., Kumar, R., & Kim, L. C. (2020, March). Blockchain meets IIoT: An architecture for privacy preservation and security in IIoT. In *2020 International Conference on Computer Science, Engineering and Applications (ICCSEA)* (pp. 1-7). IEEE.

[5] Tsimenidis, S., Lagkas, T., & Rantos, K. (2022). Deep learning in IoT intrusion detection. *Journal of network and systems management*, *30*, 1-40.

[6] Khraisat, A., & Alazab, A. (2021). A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. *Cybersecurity*, *4*, 1-27.

[7] Şahin, E., & Talu, M. F. (2022). Wy-Net: A New Approach to Image Synthesis With Generative Adversarial Networks. *Journal of Scientific Reports-A*, (050), 270-290.

[8] Jia, B., Zhang, X., Liu, J., Zhang, Y., Huang, K., & Liang, Y. (2021). Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT. *IEEE Transactions on Industrial Informatics*, *18*(6), 4049-4058.

[9] Narayanan, A., Bonneau, J., Felten, E., Miller, A., & Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.

[10] Taş, R., & Tanrıöver, Ö. Ö. (2020). A systematic review of challenges and opportunities of blockchain for E-voting. *Symmetry*, *12*(8), 1328.

[11] Kumar, T., Harjula, E., Ejaz, M., Manzoor, A., Porambage, P., Ahmad, I., Liyanage. M., Braeken. A., Ylianttila, M. (2020). BlockEdge: blockchain-edge framework for industrial IoT networks. *IEEE Access*, *8*, 154166-154185.

[12] Guo, X., Zhang, G., & Zhang, Y. (2022). A Comprehensive Review of Blockchain Technology-Enabled Smart Manufacturing: A Framework, Challenges and Future Research Directions. *Sensors*, *23*(1), 155.

[13] Abdelmaboud, A., Ahmed, A. I. A., Abaker, M., Eisa, T. A. E., Albasheer, H., Ghorashi, S. A., & Karim, F. K. (2022). Blockchain for IoT applications: taxonomy, platforms, recent advances, challenges and future research directions. *Electronics*, *11*(4), 630.

[14] Tapscott, A., & Tapscott, D. (2017). How blockchain is changing finance. *Harvard Business Review*, *1*(9), 2-5.

[15] Samuel, O., Omojo, A. B., Mohsin, S. M., Tiwari, P., Gupta, D., & Band, S. S. (2022). An anonymous IoT-based E-health monitoring system using blockchain technology. *IEEE Systems Journal*.

[16] Ray, P. P., Chowhan, B., Kumar, N., & Almogren, A. (2021). BIoTHR: Electronic health record servicing scheme in IoT-blockchain ecosystem. *IEEE Internet of Things Journal*, *8*(13), 10857-10872.

[17] Fu, X., Wang, H., & Shi, P. (2021). A survey of Blockchain consensus algorithms: mechanism, design and applications. *Science China Information Sciences*, *64*, 1-15.

[18] Uddin, M. A., Stranieri, A., Gondal, I., & Balasubramanian, V. (2021). A survey on the adoption of blockchain in iot: Challenges and solutions. *Blockchain: Research and Applications*, *2*(2), 100006.

[19] Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. *IEEE Access*, *10*, 40281-40306.

[20] Babu, E. S., SrinivasaRao, B. K. N., Nayak, S. R., Verma, A., Alqahtani, F., Tolba, A., & Mukherjee, A. (2022). Blockchain-based Intrusion Detection System of IoT urban data with device authentication against DDoS attacks. *Computers and Electrical Engineering*, *103*, 108287.
[21] Tahir, M., Sardaraz, M., Muhammad, S., & Saud Khan, M. (2020). A lightweight authentication and authorization framework for blockchain-enabled IoT network in health-informatics. *Sustainability*, *12*(17), 6960.
[22] Chen, S., Fu, X., Si, H., Wang, Y., Gao, S., & Wang, C. (2022). Blockchain for Health IoT: A privacy-preserving data sharing system. *Software: Practice and Experience*, *52*(9), 2026-2044.
[23] Senan, E. M., Al-Adhaileh, M. H., Alsaade, F. W., Aldhyani, T. H., Alqarni, A. A., Alsharif, N., ... & Alzahrani, M. Y. (2021). Diagnosis of chronic kidney disease using effective classification algorithms and recursive feature elimination techniques. *Journal of Healthcare Engineering*, *2021*.
[24] Wu, J., Zheng, D., Wu, Z., Song, H., & Zhang, X. (2022). Prediction of Buckwheat Maturity in UAV-RGB Images Based on Recursive Feature Elimination Cross-Validation: A Case Study in Jinzhong, Northern China. *Plants*, *11*(23), 3257.
[25] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321-357.
[26] Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, (3), 408-421.
[27] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, *15*(1), 1929-1958.
[28] Ozdemir, D., & Arslan, N. N. (2022). Analysis of Deep Transfer Learning Methods for Early Diagnosis of the Covid-19 Disease with Chest X-ray Images. *Düzce Üniversitesi Bilim ve Teknoloji Dergisi*, *10*(2), 628-640.
[29] Kushwaha, S. S., Joshi, S., Singh, D., Kaur, M., & Lee, H. N. (2022). Systematic review of security vulnerabilities in ethereum blockchain smart contract. *IEEE Access*, *10*, 6605-6621.
[30] Oliva, G. A., Hassan, A. E., & Jiang, Z. M. (2020). An exploratory study of smart contracts in the Ethereum blockchain platform. *Empirical Software Engineering*, *25*, 1864-1904.
[31] Azbeg, K., Ouchetto, O., & Andaloussi, S. J. (2022). BlockMedCare: A healthcare system based on IoT, Blockchain and IPFS for data management security. *Egyptian Informatics Journal*, *23*(2), 329-343.
[32] Hussien, H. M., Yasin, S. M., Udzir, N. I., Ninggal, M. I. H., & Salman, S. (2021). Blockchain technology in the healthcare industry: Trends and opportunities. *Journal of Industrial Information Integration*, *22*, 100217.
[33] Ho, Y., & Wookey, S. (2019). The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE access*, *8*, 4806-4813.
[34] Zhang, Z., & Sabuncu, M. (2018). Generalized cross entropy loss for training deep neural networks with noisy labels. *Advances in neural information processing systems*, *31*.
[35] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).
[36] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems.*