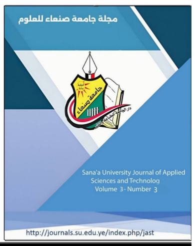
Vol.3, No.3, Page 898- 914, 2025 https://doi.org/10.59628/jast.v3i3.1681 ISSN 2958-9568 JAST

SUN PUBLISHER

مجلة جامعة صنعاء للعلوم التطبيقية والتكنولوجيا

Sana'a University Journal of Applied Sciences and Technology

https://journals.su.edu.ye/index.php/jast/



Enhanced Asynchronous Advantage Actor-Critic (EA3C) Algorithm for Intrusion Detection in IIoT Environment

Abdulssalam M. Khako [*] and Sharaf A. Alhomdy

Department of Information Technology, Faculty of Computer and Information Technology, Sana'a University, Sana'a, Sanaa, Yemen

*Corresponding author: abs-khako@hotmail.com

Abstract

Intrusion detection is crucial for securing Industrial Internet of Things (IIoT) networks, especially within edge computing environments. Traditional Intrusion Detection Systems (IDSs) struggle with the complexity and dynamic nature of IIoT networks, where increasing intrusion classes make classification tasks more challenging. While the Asynchronous Advantage Actor-Critic (A3C) algorithm has shown promise in reinforcement learning-based IDSs, previous A3C implementations suffer from slow convergence, high variance, unstable gradient updates, and inefficient parameter synchronization. These issues limit their ability to accurately classify diverse attack patterns, particularly underrepresented intrusion types. To address these challenges, this research introduces an Enhanced A3C (EA3C) using an enhanced convolutional neural network (CNN) structure to significantly improve feature representation compared to traditional fully connected networks from the dataset before passing them to the policy and value networks. Additionally, gradient clipping will be applied to prevent exploding gradients, and bootstrapping-based reward handling will be used to enhance policy and value estimation for better long-term learning and improved intrusion detection performance.

The proposed approach is evaluated using the X-IIoTID dataset, which is a comprehensive benchmark that is effective in detecting a wide range of cyber threats in IIoT environments. Experimental results indicate that the EA3C algorithm significantly outperforms Decision Tree (DT), Adversarial Environment Reinforcement Learning (AERL), Double Deep Q-Network (DDQN), and traditional A3C algorithms, particularly in identifying underrepresented attack classes. The results of EA3C show that its weighted Accuracy, Precision, Recall, and F1-score exceeded 0.98, making it suitable for practical use with the increasing number of labeled classes of cyber-attacks. Although these results are promising, this algorithm needs further improvement, especially for attacks with very small samples or attacks that occur for the first time, such as zero-day attacks.

Article Info

Keywords:
A3C, DRL, IDS, IIoT, X-IIoTID

Article History:
Received: 7-April-2025,
Revised: 15-May-2025,
Accepted: 23-May-2025,
Available online: 30 June 2025.

1. INTRODUCTION:

Cyber intrusion detection is a major security priority in IIoT networks linked to edge computing across the network. IIoT is used in edge computing to process data and make real-time decisions in several important fields such as agriculture, industry, medicine, etc., improving the responsiveness and efficiency of industrial operations [1]. However, this distributed infrastructure has flaws that cybercriminals may exploit as vulnerabilities to carry out their attacks, which may lead to huge problems [2].

These problems arise because the IIoT network contains several components such as control units, sensors, and actuators [3]. For example, any vulnerability may be exploited by hackers through the use of communication channels, systems, or protocols to gain unauthorized access. This may lead to significant damage, such as data theft, data modification, or the halt of basic operations, causing significant financial losses in the economic field [4].

Moreover, it is difficult to detect intrusion due to the nature of the end devices in the IIoT network. This is because of the limited resources in these devices, such as memory capacity and processing power [5]. Hence, the performance of intrusion detection and efficiency of these resources must be balanced in IDS in edge computing to prevent the end devices from exhausting their resources.

In addition, the IIoT network is dynamic due to the constant addition and deletion of devices and components. Traditional IoT algorithms detect attacks based on current, pre-configured settings, which may be insufficient in an environment where network components are constantly changing [6]. Therefore, IDSs need to be able to detect potentially malicious activity of changing these components.

To overcome the security problems of IIoT, several strategies can be used in IDS. IDS may be integrated with edge computing or connected to it separately due to the limited resources of IoT network devices [7]. This is done by using an artificial intelligence algorithm strategy to identify common patterns of network activity and identify data that may be suspected of being a hacking operation. This strategy increases the accuracy of detecting cyber intrusions.

Currently, actor critic algorithms are highly used in detecting cyber intrusions compared to other algorithms. One of the most popular of these algorithms is the A3C algorithm [8]. This algorithm, which outperforms other algorithms, is widely used in intrusion detection in IIoT environments because it can detect complex or abnormal intrusions due to its ability to detect rare intrusions in real time [9].

In summary, the IDS is critical to systems in networks of the IIoT environment, and the EA3C algorithm can be developed to protect data and industrial operations through the use of smart, flexible, and robust IDSs.

1.1. PROBLEM STATEMENT

The rapid expansion of IIoT has significantly increased their vulnerability to hacking. Current IDSs struggle to keep pace with the dynamic and complex nature of IIoT networks. This complexity is exacerbated by the increasing classes of attacks and the difficulty of identifying their class and dealing with them based on the behavior of these attacks [10]. Although the existing IDS algorithms, particularly those based on actor critic algorithms like A3C, have contributed to improving the classification, they still face limitations and struggles in terms of slow convergence, large variation in policy gradients, unstable updates, and inefficient parameter synchronization [11]..

1.2. MOTIVATION

As IIoT continues to expand, it is critical to develop a mechanism using artificial intelligence algorithms to detect cyber threats that could disrupt industrial operations and cause significant economic losses. Traditional IDSs based on machine learning, deep learning (DL), and reinforcement learning face several challenges in adapting to new and advanced attack patterns, given that these systems rely on pre-defined features or fixed detection models. Algorithms based on actor critic provide the ability to adapt to changes that may occur in the IIoT environment and their high self-learning capabilities. Among these algorithms is the A3C algorithm. However, the current implementation of A3C and its performance limitations affect its practical usability in IIoT security. These limitations can be improved by improving the extraction of the most important features using the fuzzy algorithm, making the training process as stable as possible, and improving policy updates, which contribute significantly to detecting attacks and identifying the classes of these attacks. Therefore, the motivation behind this research is to develop the EA3C algorithm based on CNN inside its worker/agent into a more efficient detection system based on classifying common and rare attack classes with high Accuracy, Precision, Recall, and F1-score compared to previous, more common studies, such as the DT, AERL, DDQN, and traditional A3C algorithms in IIoT environments.

1.3. OBJECTIVES

The main objective of this research is to address the challenges by leveraging advanced machine learning techniques and enhancing the A3C framework with a modified CNN as follows:

· Developing an EA3C Algorithm for enhancing Intrusion Detection in IIoT Environments.

· Evaluate the performance of the EA3C algorithm compared to DT, AERL, DDQN, and traditional A3C using metrics such as accuracy, recall, precision, and f1-score of intrusion detection in IIoT environments, to achieve more secure and resilient IIoT networks.

1.4. CONTRIBUTIONS

This research contributes significantly to the development of the field of intrusion detection in IIoT environments by enhancing the performance of A3C-based systems. The key contributions are as follows:

· **Modified Neural Network Model Architecture**: CNN architecture is introduced to significantly improve feature representation by extracting high-level spatial and temporal features from the intrusion detection data, compared to traditional fully connected A3C networks. This improvement addresses the limitations of previous feature extraction models that failed to capture complex patterns in network traffic.

· **Stabilized Training Process**: To address issues of high variance and unstable gradient updates in A3C implementations, gradient clipping is incorporated to control gradient updates, improving training stability and convergence speed.

. **Efficient Parameter Synchronization**: The proposed algorithm optimizes parameter synchronization by implementing efficient copy · operations and sorted variable lists, which enhance convergence speed and prevent delays during training.

· **Dynamic State Representation**: Unlike traditional A3C, which used fixed input representations, the EA3C algorithm dynamically reshapes state inputs to ensure proper compatibility with the CNN layers and policy/value networks, enabling more efficient policy learning. The proposed EA3C algorithm is evaluated using the X-IIoTID dataset, over existing algorithms, including DT, AERL, DDQN, and traditional A3C, using metrics of accuracy, precision, recall, and F1-score.

Through these contributions, this research offers a robust solution for intrusion detection in IIoT environments, providing faster, more stable, and higher-performing algorithms capable of addressing the evolving and complex security challenges of modern IIoT networks.

1.5. RESEARCH ORGANIZATION

The research is structured as follows: Section1 introduces the topic and emphasizes the significance of intrusion detection in IIoT networks. Section 2 provides a background, discussing existing IDS algorithms, the attacks and challenges in IIoT environments, and related works. Section 3 presents the proposed EA3C algorithm, highlighting its innovations and modifications to the A3C worker's neural network. Section 4 outlines the methodology used to develop and evaluate the EA3C algorithm, detailing the experimental setup, data preprocessing, and performance metrics evaluation. Section 5 presents the results and discusses the performance of EA3C in comparison to other algorithms, demonstrating its effectiveness. Finally, Section 6 concludes the research, summarizing the key findings and proposing future directions for further research.

2. BACKGROUND

In this section, the main concepts that support the methodology of this research will be discussed, which are as follows:

2.1. IIOT ATTACKS

Security standards in the IIoT environment are achieved through the development of effective factors such as network segmentation, upgrading hardware and software, identifying vulnerability, updating IDS and incident response plans, in addition to having a deep understanding of several classes of major attacks in this environment and their subclasses, which are mentioned in X-IIoTID dataset and are as follows:

1. Reconnaissance: It is the first stage of an IIoT attack, where the attacker uses footprinting or data collection to identify vulnerabilities in the targeted networks [12]. The subclasses of these attacks are as follows:

- Generic Scanning: where the attacker carefully searches the targeted network for open ports, versions of hardware and software in use, and potential entry points.[13].
- Scanning Vulnerability: which is a scanning process aimed at finding specific vulnerabilities in systems or IIoT devices [14].
- Discovering Resources: where attackers search for IIoT network resources. These resources include the classes of applications used, open ports, application interfaces that are usually unprotected, or improperly configured settings [15].
- Fuzzing which is defined as detecting vulnerabilities by triggering unexpected behaviors, such as sending unexpected or incorrect data to IIoT devices to test their response [16].

2. Weaponization: This is the preparation to exploit any vulnerabilities discovered through reconnaissance in the IIoT infrastructure [17]. These attacks include the use of the following subclasses:

- Brute-force: Attackers often generate a variety of usernames and passwords to gain unauthorized access to IIoT devices and systems, but this class requires more time for high processing [18].
- Dictionary: This type of attack occurs by gaining unauthorized access using pre-prepared lists of commonly used passwords, and it takes less time compared to the previous class [19].
- Insider Malicious: This attack occurs when an authorized person gains access to IIoT systems or devices by deliberately using their rights to destroy the system, steal data, or cause malfunctions [20].

3. Exploitation: This is done by interfering with unauthorized access operations by exploiting the vulnerabilities discovered in the IIoT environment[14] , which are as follows:

· Reverse Shell: Attackers create a reverse shell by exploiting flaws in IoT systems or devices due to delays in updating or upgrading them [21].

· Man-in-the-Middle (MitM): This includes the presence of attackers who intercept communications and change or eavesdrop on data in the communication channel between devices and applications of IIoT [22].

4. Lateral movement: This is used in a way that spreads the attack from one hacked system or device to another within the IIoTs network in an attempt to gain more control and benefit from the attack on the largest possible number of systems and devices that can be

accessed [23]. The following subtypes may be involved in lateral migration in IIoT attacks:

- Modbus-Register Reading: Attackers are able to obtain sensitive data from IIoT systems or devices by exploiting weaknesses in the Modbus protocols [24].
- MQTT-Cloud Broker Subscription: One popular protocol for Internet of Things (IoT) communication is Message Queuing Telemetry Transport (MQTT). By exploiting vulnerabilities in MQTT communication channels, attackers can get unauthorized access to IIoT devices or systems through cloud brokers [25].
- TCP Relay: Intercepting and rerouting TCP communication traffic between computers or IIoT devices is the tactic used in attacks utilizing Transmission Control Protocol (TCP) relays [26].

5. Command and control (C&C): Attackers establish a centralized infrastructure at the C&C stage in order to remotely control attacked IIoT devices or systems [27].

6. Exfiltration: It refers to the illegal removal or transfer of sensitive data from compromised IIoT systems or devices to other locations under the attackers' authority [28].

7.Tampering attack: It is the attack that modifies IIoT systems or data without authorization [21]. IIoT attacks may involve the following types of manipulation:

- False Data Injection: Attackers may alter or introduce fake data into IIoT systems.
- Fake Notifications: Attackers try to fool IIoT operators or users by sending fake messages or alarms.

8. Crypto ransomware: It is a type of malware that encrypts crucial information or systems, rendering them unreadable to everyone. Hackers have effectively taken control of the IIoT systems or data [29].

9. RDoS: Attackers are those who employ ransomware to prevent distributed denial-of-service (DDoS) attacks against IIoT networks or systems [30].

2.2. IIoT-IDS CHALLENGES

Researchers interested in insecurity in industrial fields have provided possible answers to these problems in recently published studies. Security and privacy concerns are the primary issues that are always taken into consideration[31] . Developing security algorithms, advanced connections, automation, and data-driven decision-making is made possible by IIoT networks at the edge of computing, which have revolutionized industrial sectors [1]. Predictive maintenance, enhanced efficiency, and process optimization are made possible by these networks, which are made up of networked machinery, devices, and sensors that collect and share enormous volumes of data. Despite the spread of IIoT networks, several security obstacles have arisen, requiring significant processing time and high power consumption. The first of these obstacles is the need to detect intrusions in the network at the edge computing level due to the dispersed and dynamic nature of this network, which consists of many interconnected devices and sensors. This network usually extends over a vast area and works in a variety of industrial environments, making the deployment of these centralized IDS represents a challenge [32]. Another obstacle, which is the immediate detection of intrusion in real time, lies in the difficulty of dealing with the enormous volume and speed of data in the IIoT network. Thus, it has become necessary to use IDSs, as it is capable of managing the continuous flow of data[33] . The last obstacle encountered in intrusion detection is that different manufacturing companies produce multiple devices and operate on various systems and communication protocols. Thus, IDSs face a challenge in managing and monitoring these components to ensure compatibility and interoperability [34].

Therefore, IDSs must be flexible enough to withstand these obstacles through using modern algorithms, such as ML, DL, DRL, and Actor Critic, that are applied in IDSs connected to edge computing to ensure their centralization and rapid response.

2.3. DEEP REINFORCEMENT LEARNING

DRL relies primarily on DL models, which contribute to RL to help agents learn to solve sequential decision-making problems. In RL, an agent interacts with its environment by maximizing the cumulative reward value by correctly predicting the input states during training [35]. The Markov decision process (MDP) is a fundamental part of the concept of RL, whose learning parameters are expressed as (S, A, P, R,γ), where (S) is the set of potential states in IIoT network traffic, (A) is the set of accessible actions, γ(gamma) is the discount factor, and (s) to state (s') is the transition probability state. Therefore, when taking action (a), it is expressed as P(s'|s, a)[36]. On the other hand, the future rewards R (s, a, s') refer to the immediate reward received by transitioning from state (s) to state (s') through the selection of action (a). The agents evaluate the probability of acting in a specific state (s) by learning a policy function π(a|s) over the possible actions. The agents/workers interact with the IIoT environment by receiving rewards and new coefficients to find the best coefficients of algorithms by maximizing rewards. At the same time, the available information can be used to achieve this using different strategies, such as softmax action selection or ε-greedy, which can also be applied. The sum of the discounted incentives obtained from a starting state (s) is used to compute the return.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{(t+k+1)} \quad (1)$$

R_{t+1}, R_{t+2}, R_{t+3},..., are the rewards at subsequent time steps, and G_t is the total of future rewards obtained starting at time step t+1 that weighs the importance of immediate versus future rewards. To find the cumulative value of the awards over time, the formula sums up the prizes and multiplies each by the corresponding discount factor γ^k. This enables decision-makers in a context including RL to consider both potential rewards in the future and current rewards. To maximize the expected return, the agent adjusts the parameters of its policy. Stochastic gradient descent (SGD) or its derivatives, like Adam, are gradient-based optimization techniques that are used to update the parameters of the deep neural network (DNN) depending on the gradients of the objective function with respect to the parameters[37]. Q-learning and Deep Q-Networks (DQN), for example, are value-based algorithms that learn to figure out state-action values and edit policies appropriately.

2.4. EXISTING A3C ALGORITHMS

Several essential components make up the standard A3C as shown in Fig.1. With numerous worker threads running concurrently and interacting with various instances of the environment and neural network algorithm, it employs asynchronous training [38].

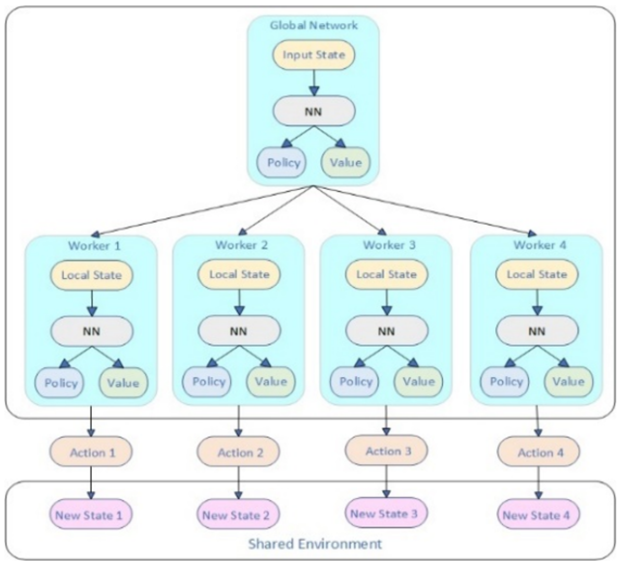


Figure 1. The conventional A3C diagram

For the actor and critic structures, the majority of recent work has used simple neural networks (NNs). Exploration and experience gathering become more efficient as a result. Moreover, A3C employs an actor-critic, where the actor component is a policy network that selects actions based on the current state [39]. The critical component, which is represented by a value network, determines the expected return from the current situation. Additionally, A3C offers advantage estimate [40], which improves training stability by comparing the benefit of doing a particular action in a state to the state's average action value. Lastly, the mean squared error loss and the policy gradient technique are used to update values and policies, respectively.

Synchronized updates are sometimes performed by averaging gradients from several worker threads to update common algorithm parameters in order to guarantee consistency and reduce conflicts. When combined, these components enable A3C to effectively train DRL agents in complex environments with high-dimensional state spaces.

2.5. (X-IIoTID) DATASET

The X-IIoTID dataset is a meticulously created simulation of current attackers' methods, techniques, and strategies as well as the actual operations of IIoT systems.

Some examples of these activities include the interaction of industrial devices (sensors, actuators, controllers, and traffic from edge, mobile, and cloud sources). The dataset reflects the behaviors of new communication protocols (e.g., MQTT, CoAP, WebSocket), the different connection patterns (Machine-to-Machine, Human-to-Machine, and Machine-to-Human), and large amounts of network traffic and system events. This is due to its ability to adjust to the heterogeneous nature and interoperability requirements of IIoT systems, in addition to its features that are independent of IIoT devices and networks.

The final X-IIoTID dataset has 820,834 cases with 68 characteristics (normal and attack, normal and subcategory attack, and normal and sub-sub-category attack encompassing three label levels) (421,417 observations/instances for normal and 399,417 for attacks). [41].

2.6. RELATED WORK

In recent years, several studies have focused on using (DL) and RL for IDS, each proposing different approaches to address challenges like anomaly detection and attack classification. Shone et al. [42]proposed a DL-based intrusion detection model that employs an unsupervised feature learning approach through a non-symmetric deep autoencoder (NDAE). Their model is built on TensorFlow with GPU acceleration to utilize a stacked NDAE-based classification algorithm for obtaining effective performance on the KDD Cup'99 and NSL-KDD datasets.

Shone et al. [43] also investigated various ML algorithms for attack classification, including DT, Naive Bayes (NB), K-Nearest Neighbors (KNN), Support Vector Machine (SVM), and Logistic Regression (LR), alongside DL techniques like Gated Recurrent Units (GRU) and DNN. Their results confirmed that the DT algorithm showed better performance results compared to other algorithms

Table 1. Related Works Summary

Ref.	Publisher	Approach	Work Points	Limitations	Dataset
[42]	IEEE transactions on emerging topics in computational intelligence	Nonsymmetric deep autoencoder (NDAE) for unsupervised feature learning	- Using simple applications - Efficient and steady sampling	- Unsuitable for a stochastic environment	KDDCup 99
[43]	IEEE Internet of Things Journal	DT	- It is simple and very fast to get the prediction results	- The performance values decrease with an increasing number of classifications	X-IIoTID
[44]	IEEE Access	DNN	- Used for simple applications - Enhanced performance compared to some supervised and unsupervised methods.	-Unsuitable for complex applications - Having issues in the real environment	KDDCup 99
[45]	IEEE Access	Recurrent Neural Network (RNN)	- For simple applications.	- Unsuitable for a stochastic environment - Having limitations in the acceleration process	NSL-KDD
[46]	IEEE Access	Novel Two-Stage DL (TSDL) model based on a stacked auto-encoder with a soft-max classifier	- Used for simple applications - Efficient and steady sampling	- Unsuitable for a stochastic environment - Unsuitable for complex applications	KDD99 and UNSW-NB15
[47]	Wireless Communications and Mobile Computing	RL to improve the decision-making ability	- Suitable for both consistent and stochastic applications	- It is considered an unstable algorithm and takes more time to process	Dataset of the natural gas pipeline transportation network
[48]	Computers	Q-learning based on reinforcement learning with a deep feed-forward neural network method	- For normal applications - Enhanced performance by taking advantage of value-based and policy-based methods	- Unsuitable for a stochastic environment -It has a lower accuracy value than actor critic algorithm.	NSL-KDD
[9]	Entropy	Adaptable A3C algorithm for reinforcement learning with CNN	- Suitable for complex applications	- It is too complex and has processing overhead	NSL-KDD, AWID, and CICIDS-2017, DoHBrw-2020

through training and testing it in different scenarios.

Vinayakumar et al. [44]built a DNN model for learning through hidden layers to extract meaningful features and optimize network topology and hyperparameters using the KDD Cup 99 dataset. This model represents high-dimensional features from network data by transferring it to the IDS.

Yin et al. [45] explored the use of Recurrent Neural Networks (RNN) for intrusion detection in IDS, which showed strong performance in detecting various attack patterns in an IIoT environment. Their model emphasized the temporal dependencies of data flow within the network, which facilitated the detection of attacks.

Khan et al. [46] proposed a two-stage DL approach for network intrusion detection, which involves a stacked autoencoder layer with a softmax layer for classification. Based on a probability score, their model classifies data from the normal and abnormal classes. The UNSW-NB15 and KDD99 datasets were used for testing to prove the effectiveness of this model.

Tharewal et al. [47] implemented a DRL algorithm to connect intrusion detection in IIoT networks. This algorithm has proven its ability to transform large-scale input data into abstract representations, which increases the algorithm's ability to make decisions without any external guidance.

Alavizadeh et al [48] combined a deep Q-learning-based RL approach with a deep feed-forward neural network for network intrusion detection. Thus, their model automatically identifies various classes of intrusions and continually improves the detection performance. The NSL-KDD dataset was used to train, test, and validate the model.

Zhou et al.[9] applied a convolutional network for anomaly detection with an attention mechanism for distinguishing between anomalies to be used as an adaptive actor-critic model. The model outperforms traditional methods on benchmark datasets like NSL-KDD, AWID, CICIDS-2017, and DoHBrw-2020.

Caminero et al. [49] presented an RL approach for

IDS based on adversarial environments. Their AERL algorithm simulates an adversarial setup where the classifier, as the primary agent, continuously faces more challenging predictions posed by the environment, ultimately improving the classifier's robustness.

The studies summarized above highlight the diverse methods employed in the field of intrusion detection, with advancements in ML, DL, and RL techniques such as DT, AERL, DDQN, and traditional A3C, aiming at improving performance and adaptability in detecting and mitigating network attacks. The related works, which were previously mentioned, are summarized in Table 1.

3. PROPOSED EA3C

The EA3C framework shown in Fig.2 uses a modified CNN that replaces the neural network in the worker of the conventional A3C algorithm. The modified CNN model,

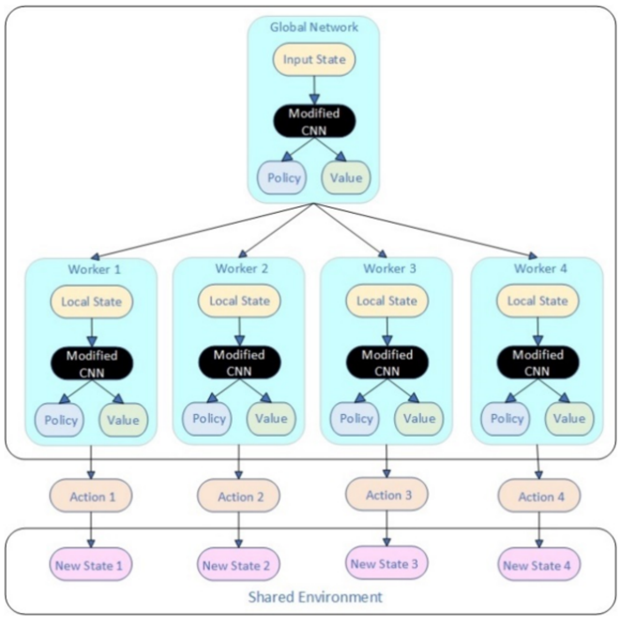


Figure 2. The modification of EA3C workers by using CNN

as shown in Fig.3, is used to automatically extract discriminative characteristics from the X-IIoTID dataset. To find patterns and relevant information specific to IIoT intrusion detection tasks, convolutional layers extract spatially local features. Through the use of strategies such as max pooling, the pooling layers reduce the dimensionality of the learned features, highlighting the most crucial components and removing irrelevant data. This enhances the algorithm's ability to identify incursions across a lot of settings.

The model first checks that X is a 2-dimensional tensor, reshaping it to include a singleton dimension. Then, it builds a network: convolutional layers with filters f = 32, kernel size k = 3, padding p = 1, ReLU activations, and batch normalization, followed by max pooling layers with a pool size ps = 2 and strides st = 2. The output is then flattened and passed through two fully connected, or dense, layers with ReLU activations and batch normalization. Added to this is a dropout layer for regularization with the dropout rate r =0.5. Finally, the final network output is made from the last dense layer, with units u = 256. Convolutional layers capture local patterns in the

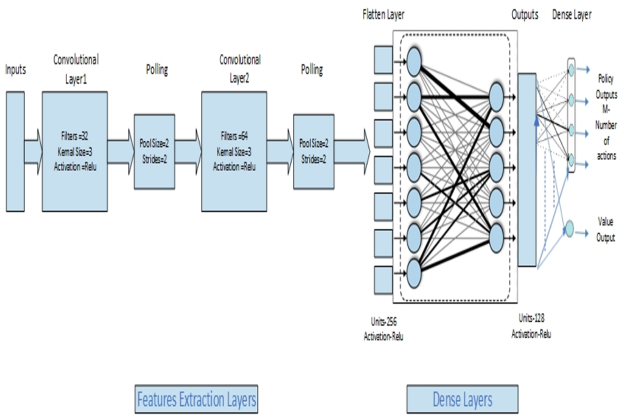


Figure 3. The modified CNN model in the EA3C worker

network traffic that help to understand network behaviors, while max-pooling layers reduce size and enforce spatial hierarchies between significant features [50]. This maintains the stability of training while applying batch normalization. Extracted features are prepared for fully connected layers using the flattening layer that subsequently learns those more complex feature relationships to efficiently classify intrusions. Dropout regularization is applied to avoid overfitting, thus ensuring the model is generalized well. This kind of architecture is good for identifying intrusion patterns in time series data, like network traffic, which leads to good accuracy and fast computing. With two convolutional layers, two pooling layers, one flatten layer, and two fully connected layers, the CNN can handle the input data effectively. High-dimensional and complicated features that are extracted from IIoT settings are often the input data.

Traditional A3C implementations often experience gradient explosion due to high-variance network updates, making training unstable where its rule is $\theta' = \theta - \alpha \nabla_\theta L$ [51]. EA3C expresses gradient clipping at a global norm of 5.0, ensuring more stable updates. The updated rule is given by Eq. 2:

$$\theta' = \theta - \alpha \cdot \text{clip}\left(\nabla_\theta L, -\tau, \tau\right) \tag{2}$$

where θ represents the model parameters, α is the learning rate, $\nabla_\theta L$ is the gradient of the loss function, and τ=5.0 ensures stable updates by preventing excessively large gradient steps. The *clip* scales the gradient $\nabla_\theta L$ by dividing it by a factor of $1 + \|\nabla_\theta L\|^2$, which helps stabilize learning by preventing excessively large updates. This technique can improve convergence and robustness, especially in RL and DL models. In traditional A3C

[52], policy updates are commonly derived using policy gradient theory as described in Eq.3

$$\nabla_\theta J(\theta) = \mathrm{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\left(R_t - V\left(s_t\right)\right)\right] \quad (3)$$

The gradient update is modified with additional constraints to stabilize training according to Eq.4

$$\nabla_\theta J(\theta) = \mathrm{E}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta\left(a_t \mid s_t\right)\left(R_t - V\left(s_t\right)\right)\right] + \lambda \nabla_\theta \Omega(\theta) \quad (4)$$

This equation represents the policy gradient in actor-critic methods. The term $\nabla_\theta J(\theta)$ denotes the gradient of the objective function

$(J\theta)$ concerning policy parameters θ. The expectation $E[.]$ is taken over trajectories generated by the policy π_θ. The summation from $t = 0$ to T accounts for all time steps in an episode. The term $\nabla_\theta \log \pi_\theta(a_t|s_t)$ represents the gradient of the log probability of taking action a_t state s_t under policy π_θ, which helps to adjust the policy toward better actions. The advantage function $(R_t - V(s_t))$ measures how much better the actual return R_t is compared to the estimated value function $V(s_t)$, guiding the policy update. The last term, $\lambda \nabla_\theta \Omega\theta$, is a regularization term controlled by λ to prevent overfitting and ensure smooth policy updates.

Step sizes are dynamically adjusted in EA3C to optimize learning efficiency, ensuring that state-action representations remain valid. This adaptive mechanism enhances exploration and improves sample efficiency, leading to more robust policy learning.

The updated value loss function L_V used in EA3C is expressed in Eq.6, where the first term is

$$L_v = \sum_{t=0}^{T} \left(R_t - V\left(s_t\right)\right)^2 \quad (5)$$

Eq.5 is used in all actor critic methods [53], sums over all time steps t in an episode, and calculates the squared difference between the estimated return reward R_t (computed via bootstrapping method) and the value function $V\left(s_t\right)$, which predicts the expected future reward from state s_t. This term ensures that the critic (value function) learns accurate state-value estimates. The second term, as in Eq.6, $\beta\|\theta\|^2$, is an $L2$ regularization term weighted by β, which is added to the value loss equation of EA3C, and helps to prevent overfitting by penalizing large policy parameter θ values. This term ensures smoother updates and better generalization.

$$L_v = \sum_{t=0}^{T} \left(R_t - V\left(s_t\right)\right)^2 + \beta\|\theta\|^2 \quad (6)$$

Each worker thread interacts with the environment autonomously, adhering to a set of rules of its own. A worker gathers environmental observations at each time step, inputs them into its CNN, and extracts the current state representation s_t. Once the worker completes a predetermined number of time steps, they create experiences in the form of triples (s_t, a_t, r_t), representing state, action, and reward. These interactions are kept in a common replay buffer for further use. Algorithm 1 gives the optimization of policy and value functions in the EA3C worker, which includes entropy regularization for better exploration. It defines all this in advance: states, targets, and actions, and then some network parameters. It also defines how many possible actions there are and the dimensionality of the observation space. The states are normalized and fed into a shared neural network, returning outputs used for both the policy and value networks. The policy network calculates the probabilities of actions using a softmax function of a dense layer, and the value network does the prediction of the value for each state. Entropy is calculated for the probabilities of actions to encourage exploration; policy loss is then computed by combining log-probabilities for the chosen actions with targets and entropy, modulated by a factor β. The value loss is the mean squared error between predicted and actual values. On each iteration, if the training flag is set, it will update the network parameters using the Adam optimizer and then proceed until it reaches a maximum number of iterations. The ε-greedy exploration strategy is used with a decay to slowly reduce exploration during training (a very common approach, for which an adaptive way is proposed to adjust the exploration rate).

This methodology contrasts with conventional A3C implementations by using more complex network architectures, entropy-based policy regularization, and a more sophisticated exploration procedure. This is designed to increase efficiency in learning and overall algorithm stability, allowing it to better adapt to high-dimensional and complex environments with an increasing number of intrusion classes.

To keep CNN and its local algorithm synchronized, the worker updates its weights and gradient values regularly. By doing this, the proposed algorithms are more closely matched and divergent learning is prevented. This CNN is modified by an asynchronous algorithm that periodically samples a batch of experiences from the replay buffer. The gradients based on the sample experiences are evaluated by the worker thread, which subsequently applies them to the proposed algorithm. The update of asynchronous processes allows for better exploration, faster convergence, and simultaneous training.

Algorithm 3.1 Algorithm for policy and value predictions in EA3C

```
1:  Initialization:
2:  s∈R$^{N\times F}$:States, a∈R$^{N}$:Actions
3:  M: Number of possible actions.
4:  y $\in$ R$^{N}$: Targets (discounted rewards)
    tr$\in\{$True,False$\}$: Flag to add training operations.
5:  $\varepsilon$: Exploration rate (for $\varepsilon$-greedy policy)
6:  $\theta_v$ , $\theta_p$                      // Initialize value and policy weights
7:  g$_{step}$ :=0 and N:=0
8:  Normalize Inputs: X := Normalize(s)        // Min-Max scaling
9:  N$_{max}$: Maximum number of steps per episode
10: $\beta$=0.01 // Entropy regularization coefficient
11: $\gamma$: Discount factor for reward calculation
12: shared_network_params: Predefined architecture for shared neural network

13: while N$<$ N$_{max}$ do
14:         # Build the shared CNN architecture:
15:         out:=SharedCNN(X)
16:         #Policy Network:
17:         $\pi_s$=Dense(out, u=M)              // Output size
18:         equals number of actions (M)
19:         P=softmax($\pi_s$ +10e$-$8)        // Softmax with numerical stability
20:         predP=$\{$ $\pi_s$ , P$\}$           // Predicted probabilities for actions

21:         #Value Network:
22:         Vs=Dense(out,u=1)        // Predicts state-value (V) for input
23:         state
24:         Vs $\in$R$^{N\times 1}\Longrightarrow$ Vs $\in$R$^{N}$        // Reshaping Vs to match the
25:         batch size (N)
26:         predV=$\{$ Vs $\}$                  // Predicted value of the current
27:         state

28:         #Entropy Calculation:
29:         H=$-\sum$P$\cdot$log(P)            // Entropy of the policy (measuring
30:         exploration)
31:         Entropy = mean(H)            // Mean entropy over all actions
32:
33:         #Policy Loss Calculation:
34:         indices=$\{$i$\times$M+Ai$|$i$\in\{$0,1,$\ldots$,N$-$1$\}\}$ // Index of selected actions
35:         picked=P[indices]            // Probability of selected actions
36:         L=$-$(log(ppicked)$\cdot$y+$\beta\cdot$H)          // Combined policy loss with
37:         entropy
38:         L$_p$=$\sum$L                            // Sum of policy losses for all
            steps
39:
40:         #Value Loss Calculation:
41:         $\delta_t$ = r$_t$ + $\gamma$V(s$_{t+1}$, $\theta_v$) - V(s$_t$,$\theta_v$)   // the temporal difference (TD)
42:         error
43:         losses= $(\delta_t)^2$          // Mean squared error for value prediction
44:         Lv =$\sum$losses                // Sum of value losses over all steps
45:
46:         # Total Loss:
47:         L$_{total}$ = Lp + Lv        // Combined policy and value loss

48:
            # Exploration strategy:
49:         if random() $<\varepsilon$: // $\varepsilon$-greedy: with probability $\varepsilon$, take random
            action
50:         a = random_action()
51:         else:      // Otherwise, take action with the highest probability
52:         a = argmax(P)
53:
54:         # Apply action and step to the next state:
55:         S$_{t+1}$, tr = step(predV, predP)
56:
57:         if tr is True and L$_{total}$ $>$ threshold then:
58:                 optimizer=AdamOptimizer(learning_rate=0.0002)
59:                 grads,vars=$\{(\nabla\theta_i$L$_{total}$,$\theta_i)|$i=1,2,$\ldots$,n$\}$ // Compute
60:                 gradients of the total loss
61:                 optimizer.apply_gradients(grads, vars) // Apply policy
62:                 and value gradients to update the model weights
63:                 $\theta_v$ :=$\theta_v$+$\alpha\nabla\theta_v\delta_t$ // Update model parameters using
64:                 value gradient descent
65:                 $\theta_p$ := $\theta_p$ + $\alpha$ $\nabla$ log $\pi$(a$_t$|s$_t$, $\theta_p$) $\delta_t$  // Update model
66:                  parameters using policy gradient descent
                     g$_{step}$:= g$_{step}$ +1 // Increment global step counter
67:         end if
68:
69:          N := N + 1 // Increment step count
              $\varepsilon$ := max($\varepsilon$_min, $\varepsilon$ * decay_rate)
    end while
```

4. METHODOLOGY

The methodology used to develop and evaluate the proposed EA3C algorithm in this research for detecting intrusions in the IIoT environment consists of a set of phases, and each phase consists of a set of steps. These phases and their steps, as shown in Fig.4 , start from data pre-processing and end with performance evaluation, which ensures the reliability of its effectiveness in detecting intrusions.

4.1. DATA PREPROCESSING

Data preprocessing is an essential phase in preparing the dataset for training and testing the algorithm. This stage includes selecting a suitable dataset, filtering the data, selecting the most important features, and handling class imbalances to ensure optimal performance for this algorithm.

4.1.1. DATASET SELECTION

The X-IIoTID dataset was selected in this research because it contains traffic data that is considered relevant to IIoT intrusion detection and is taken from a real-world environment, compared to other datasets. This dataset includes various classes of cyber-attacks, such as Distributed Denial-of-Service (DDoS), Denial-of-Service (DoS), Reconnaissance attacks, and so on, which makes it more suitable for performance evaluation of intrusion detection algorithms.

4.1.2. DATA CLEANING

Several data cleaning steps were performed in the data preprocessing phase to ensure the integrity and quality of the dataset:

Handling missing values: Missing data is compensated with -1 was either removed or imputed using statistical techniques to avoid biases in algorithm training.

Outlier detection and removal: The Interquartile Range (IQR) method was used to detect and eliminate extreme values that could affect algorithm performance.

Duplicate removal: Redundant records were

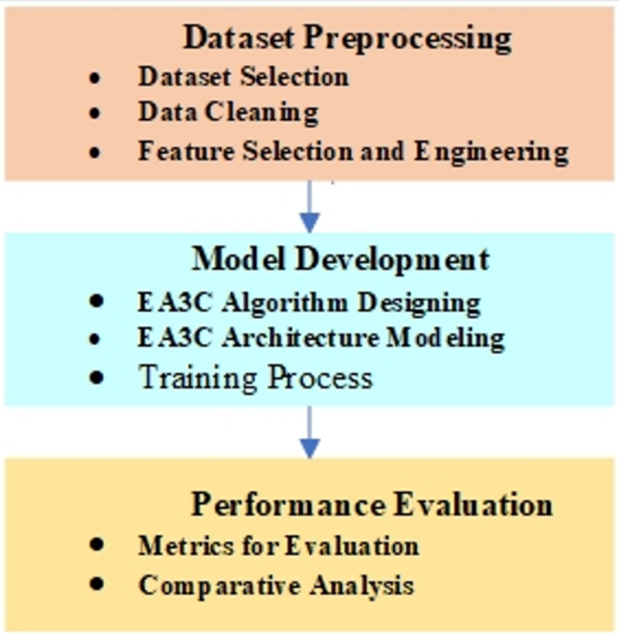


Figure 4. Research Methodology

identified and removed to prevent overfitting and ensure an unbiased learning process.

4.1.3. FEATURE SELECTION AND ENGINEERING

The 20 most important features are selected to help EA3C reduce computational costs and improve algorithm accuracy. The research employed the following techniques:

Correlation Analysis: Pearson's correlation coefficient was used to identify highly correlated features, eliminating redundancy.

Feature Scaling: Min-Max Scaling was applied to normalize continuous variables, ensuring a uniform range between 0 and 1.

Categorical Encoding: One-hot encoding and label encoding were used to convert categorical variables into a format that the neural network could process effectively, such as IP addresses and protocols. Feature Selection: A Fuzzy algorithm is used to select the best 20 features from the dataset as explained in Algorithm 2

Algorithm 4.1 Fuzzy System Algorithm for Top 20 Features Selection

```
1:  $X = [x_1, x_2, ..., x_n] \in \mathbb{R}^{m \times n}$ // A matrix of $m$ instances and $n$ features
2:  $y = [y_1, y_2, ..., y_m]^T \in \mathbb{R}^m$ // The target vector
3:  $F_i = \sigma^2$ between $(X_i)/\sigma^2$ within $(X_i)$ // Compute the ANOVA F-score for
  each
4:  feature $F = [F_1, F_2, ..., F_n]$ Vector of F-values
5:  $Q_i = \min(F_i/\max(F) \times 10,\ 10)$ // $Q_i \in [0, 10]$
6:  Rule 1: IF $Q_i$ is good THEN $I_i$ is good
7:  Rule 2: IF $Q_i$ is average THEN $I_i$ is average
8:  Rule 3: IF $Q_i$ is poor THEN $I_i$ is poor
9:  $I_i = (\int_a^b x \cdot \mu_i(x)dx)/(\int_a^b \mu_i(x)dx)$ //Compute the crisp output importance
10: $I = [I_1, I_2, ..., I_n]$ //The final vector of importance scores
11: // Define the selected top-k features as:
12: TopFeatures $= \{x_{j_1}, x_{j_2}, ..., x_{j_k}\}$ where $j_1, ..., j_k \in$ argsort$(I)[-k:]$ // Selecting
13: top 20 features):
```

The process of feature selection begins with ANOVA F-statistics to evaluate each feature xi from a dataset. The F-score for x_i is computed as F_i = (Variance Between Classes forx_i) / (Variance Within Classes for x_i), yielding a score vector F=[F_1,. . . ,F_n]. These scores are normalized to [0,10] via Q_i, which represents feature quality. A fuzzy inference system then maps each Q_i to an importance score Ii using rules (e.g., "IF quality is good THEN importance is good") and membership functions ("poor," "average," "good"). Mamdani inference with centroid defuzzification computes I_i. Finally, features are ranked by I=[I1,. . . , In], and the top-k features (here, k=20) are selected and sorted as the top 20 features as shown in Fig.5.

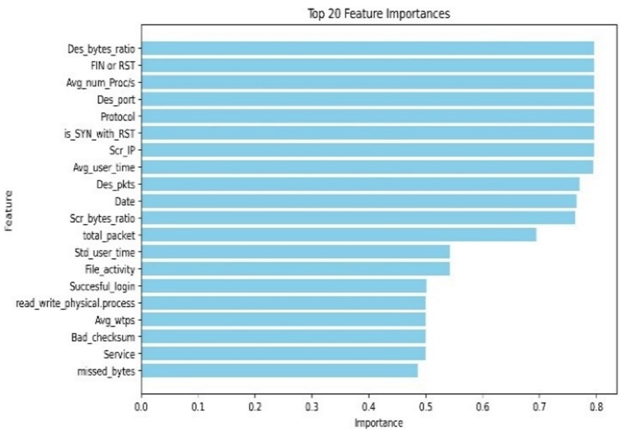


Figure 5. The best 20 selected features

4.1.4. DATASET SPLITTING

In X-IIoTID dataset, the data has been organized and distributed based on both high-level attack classes (10 classes) and their corresponding detailed sub-classes (19 sub-classes). Each attack class, such as Reconnaissance, Weaponization, Exploitation, Lateral Movement, Command and Control, Exfiltration, Tampering, CryptoRansomware, and RDoS, along with the Normal class, was broken down into its specific sub-attack classes (e.g., Scan-

Table 2. The distribution of each class in X-IIoTID dataset

No.	10 Classes	19 Sub-Classes	Total Samples	Training (70%)	Testing (30%)
1	**Normal**	**Normal**	421,417	295,012	126,405
2	**RdoS**	**RDOS**	141,261	98,883	42,378
3	**Reconnaissance**	**Scanning vulnerability**	52,852	37,000	15,852
4		**Generic_scanning**	50,277	35,194	15,083
5		**BruteForce**	47,241	33,068	14,173
6		**MQTT_cloud_ broker_subscription**	23,524	16,467	7,057
7	**Weaponization**	**Discovering_resources**	23,148	16,203	6,945
8		**Exfiltration**	22,134	15,494	6,640
9		**insider_malicious**	17,447	12,213	5,234
10	**Exploitation**	**Modbus_register_ reading**	5,953	4,167	1,786
11		**False_data_injection**	5,094	3,566	1,528
12	**Lateral_Movement**	**C&C**	2,863	2,004	859
13		**Dictionary**	2,572	1,800	772
14		**TCP Relay**	2,119	1,483	636
15	**Command_and_Control**	**fuzzing**	1,313	919	394
16	**Exfiltration**	**Reverse_shell**	1,016	711	305
17	**Tampering**	**crypto-ransomware**	458	321	137
18		**MitM**	117	82	35
19	**CryptoRansomware**	**Fake_notification**	28	19	9
		Total	**823,834**	**576,684**	**247,150**

ning_vulnerability, BruteForce, Reverse_shell). The total number of samples for each subclass was calculated, and then the number of samples for each class was collected. After that, the total number for all these classes was determined, and then each class was divided into 70% for training and 30% for testing. This ensures a logical distribution for all classes and then contributes later to the proposed EA3C algorithm that enhances the reliability of the performance of classifying common attacks in the IIoT network.

4.2. EA3C DEVELOPMENT

EA3C development focuses on designing and implementing the EA3C algorithm to improve intrusion detection in IIoT environments.

4.2.1. ENHANCED A3C ALGORITHM

The proposed EA3C algorithm is an improved version of the A3C algorithm, specifically designed to handle the stochastic nature of intrusion detection problems. Unlike traditional A3C algorithms, EA3C introduces architectural modifications that enhance learning efficiency and classification accuracy.

4.2.2. EA3C ARCHITECTURE

The EA3C algorithm consists of three main components:

· Actor Network: It is responsible for policy learning and intrusion classification.

· Critic Network: It evaluates actions and provides feedback to refine learning.

· Advantage Function: It helps reduce policy variance and stabilizes learning.

· To enhance performance, the following modifications were introduced:

Modified CNN in agent/worker as shown in Fig.6. This model improves classification accuracy with various optimizations. It consists of an input layer for raw data, followed by convolutional layers that extract hierarchical features using techniques including grouped convolutions (making deeper networks more effective), activation functions Leaky ReLU, pooling layers (reduce spatial dimensions), batch normalization (stabilizes learning), dropout layers (prevent overfitting), and fully connected layers (convert extracted features into predictions using softmax outputs). Optimizations include advanced loss functions (Cross-Entropy) and adaptive optimizers (Nadam). · Residual

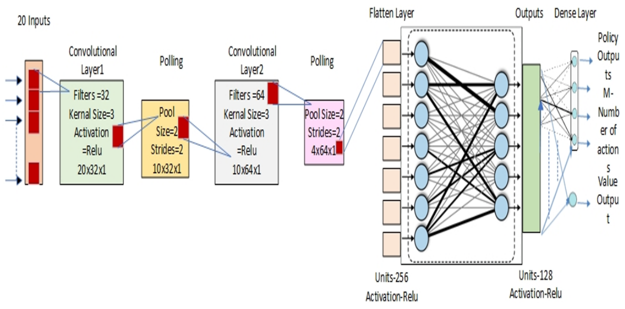


Figure 6. Modified CNN diagram in A3C agent/worker

Connections: Helps prevent vanishing gradient issues, ensuring stable training over multiple layers.

4.2.3. *TRAINING PROCESS*

The EA3C algorithm was trained in an asynchronous multi-threaded environment, where multiple workers processed different dataset batches simultaneously.

Advantage Function optimization: This stabilized action selection, reducing erratic policy updates.

Adaptive Learning Rate Scheduling: It dynamically adjusts the learning rate to prevent overfitting and enhance generalization.

Parallel Processing: Multiple training threads (8 Threads) improved computational efficiency and reduced convergence time.

4.3. PERFORMANCE EVALUATION

Performance evaluation is essential to assess the effectiveness of the EA3C algorithm compared to baseline models and algorithms. The evaluation includes classification metrics and computational efficiency analysis.

4.3.1. *METRICS FOR EVALUATION*

The effectiveness of the EA3C algorithm in this research is evaluated using the main performance metrics: accuracy, precision, recall, and F1 score. These metrics provide a comprehensive assessment of the algorithm's classification performance, ensuring a fair assessment of its ability to correctly identify positive and negative cases.

4.3.2. *COMPARATIVE ANALYSIS*

The EA3C model was compared against DT, AERL, DDQN, and Traditional A3C. EA3C outperformed all these algorithms, particularly in detecting underrepresented intrusion classes. The results showed that EA3C achieved over 98% accuracy, precision, recall, and F1-score. These results ensure its superior ability to classify complex IIoT attacks.

5. RESULTS AND DISCUSSION

The EA3C algorithm was developed using TensorFlow, as is the case with most ML, DL, and DRL investigations. All tests were conducted using TensorFlow with GPU support, an AMD Radeon (TM) R5 M330 GPU running 64-bit Windows 10, an Intel Core i5 7th Generation 2.70GHz CPU, and 16 GB of RAM. The X-IIoTID dataset has been used for the evaluations. This dataset is considered a benchmark for IIOT-IDS research. Comparing results and methodologies with earlier research is also made simpler by using this dataset.

This section makes use of the following parameters: 1) A True Positive (TP) is attack data that is correctly classified as an attack. 2) False Positive (FP): Information that is wrongly classified as an attack but is regarded as normal. 3) True Negative (TN): Normal data that is correctly classified as such. 4) A false negative (FN) is attack data that has been incorrectly classified as normal. The effectiveness of this proposed method is evaluated using the Accuracy, Precision, Recall, and F1-score criteria.

The accuracy calculates the percentage of all correctly classified cases, as in Eq 5:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7}$$

The precision is a metric that counts how many correct classifications are penalized for every wrong classification, as in Eq. 6:

$$Precision = \frac{TP}{TP + FP} \tag{8}$$

The recall calculates the number of accurate categories that are deducted from the total number of incorrect inputs, as in Eq. 7:

$$Recall = TP/(TP + FN) \tag{9}$$

The F1-score (F1) is a derived effectiveness metric that calculates the harmonic mean of recall and accuracy, as in Eq. 8:

$$F1 - score = 2(PrecisionRecall)/(Precision + Recall) \tag{10}$$

Table2 presents the average performance findings, which were trained and tested 100 times for 10 experiments per algorithm. With a train-test split (70/30 split) for X-IIoTID dataset. When compared to other algorithms, it is clear from the table that EA3C algorithm delivers improved accuracy, precision, recall, and F1-score (F1) with increasing the number of classes. The results presented in this table highlight the performance of five algorithms, which are DT, AERL, DDQN, A3C, and EA3C, across three classification tasks: Binary, 10-Class, and 19-Class Classification. decreased in accuracy (99.54%), precision (94.34%), and recall (94.52%), leading to a decrease in F1 score (94.93%). This indicates that DT struggles to perform as the number of classes increases. The DDQN algorithm performs well (95.43%), while the AERL algorithm has limitations in complex classification tasks, with an accuracy of 89.16%.

EA3C demonstrates its robustness and effectiveness in handling complex classification problems by consistently outperforming all other models and algorithms on all classification tasks. A3C also performs exceptionally well,

In the binary classification task, DT achieves the highest performance (99.57%) across all metrics, indicating its strong capability in distinguishing between the two classes. EA3C follows closely with 98.51%, demonstrating the advantage of actor cretic algorithms over traditional DT. A3C (98.01%) and DDQN (98.14%) also perform well, though slightly below EA3C. AERL lags with 97.47% accuracy, showing a noticeable performance gap compared to other algorithms. The results suggest that EA3C is highly effective, while DT remains a strong baseline in binary classification. As the classification task becomes more complex with 10 classes, the performance of some algorithms starts to diverge. EA3C outperforms all algorithms with an accuracy of 99.66%, demonstrating its ability to generalize well to multi-class problems. A3C follows closely with 99.46%, while DT, which performed best in binary classification, drops slightly to 99.56% accuracy, indicating its relative struggle in handling multiple classes. DDQN achieved an accuracy of 96.04%, outperforming AERL, which dropped to 92.86%. This is confirmed by the precision, recall, and F1 score values. These results demonstrate that the actor-critical models (A3C and EA3C) maintain strong performance compared to AERL, which struggles to adapt effectively. In the most challenging classification task, which includes 19 classes, EA3C's superiority is achieved in terms of the highest accuracy (99.85%), precision (98.60%), recall (98.58%), and F1 score (98.58%). Also, A3C followed with an accuracy of 99.71%, maintaining high performance metrics. The DT showed excellent results in binary and decimal classifications, with an increasing number of classes, its performance metrics decreased in accuracy (99.54%), precision (94.34%), and recall (94.52%), leading to a decrease in F1 score (94.93%). This indicates that DT struggles to perform as the number of classes increases. The DDQN algorithm performs well (95.43%), while the AERL algorithm has limitations in complex classification tasks, with an accuracy of 89.16%. EA3C demonstrates its robustness and effectiveness in handling complex classification problems by consistently outperforming all other models and algorithms on all classifi-

Table 3: Performance results (%)

Algorithm	Binary Classifications				10 Classifications				19 Classifications			
	A	P	R	F1	A	P	R	F1	A	P	R	F1
DT	**99.57**	**99.57**	**99.57**	**99.57**	99.56	97.44	97.34	97.29	99.54	94.34	94.52	94.93
AERL	97.47	97.67	97.46	97.52	92.86	94.65	92.87	93.40	89.16	91.80	89.16	89.90
DDQN	98.14	98.16	98.15	98.15	96.04	95.51	96.05	95.71	95.43	95.28	95.43	95.15
A3C	98.01	98.06	98.01	98.03	99.46	97.36	97.32	97.31	99.71	97.17	97.24	97.16
EA3C	98.51	98.53	98.51	98.51	**99.66**	**98.34**	**98.34**	**98.34**	**99.85**	**98.60**	**98.58**	**98.58**

cation tasks. A3C also performs exceptionally well, particularly in higher-class classifications, highlighting the effectiveness of RL techniques in multi-class scenarios. DT, while highly effective in binary classification, struggles as the number of classes increases, making it less suitable for complex tasks. DDQN provides balanced performance across different classification tasks, whereas AERL exhibits the weakest results, particularly in multi-class settings. These results highlight the advantage of actor critic algorithms such as A3C and EA3C in handling complex, 19-class classification problems. EA3C's superior performance indicates that enhancing A3C with additional optimizations leads to improved learning efficiency and generalization across diverse classification scenarios. Fig.7 presents the Receiver Operating Characteristic (ROC) analysis, which visually represents the trade-off between true positive rate (sensitivity) and false positive rate for both the A3C and EA3C algorithms. The ROC curve is a crucial evaluation tool, as it highlights the classification performance across different threshold settings. The results indicate that EA3C outperforms A3C, with a noticeable improvement in accuracy from 99.71% to 99.85%. This suggests that EA3C not only achieves higher overall accuracy but also demonstrates superior discrimination capability in distinguishing between different classes. The ROC curve for EA3C is likely to be closer to the top-left corner, indicating a higher true positive rate with minimal false positives. In contrast, A3C, while still achieving high performance, has a relatively lower accuracy, suggesting that it may produce more misclassifications compared to EA3C. The training performance of EA3C over a 24-hour duration, as illustrated in Fig.8 , demonstrates a steady and consistent increase in the reward mean, ultimately peaking at 0.9875. This sustained improvement indicates that EA3C not only achieves superior learning efficiency but also maintains stability throughout training. Unlike the traditional A3C algorithm, which may experience fluctuations or slower convergence due to parameter inconsistencies, EA3C benefits from enhanced network synchronization, ensuring more reliable updates across workers. Additionally, gradient clipping and multi-step training contribute to a smoother learning process, preventing instability and reducing variance. The robust action selection mechanism further ensures that decisions remain well-optimized as training progresses. The increasing reward trend confirms EA3C's ability to efficiently explore and exploit the learning environment, leading to improved long-term performance. This makes EA3C more effective than the RL algorithm, particularly for complex classification tasks where stability and convergence speed are critical. The confusion matrix, as shown in Fig.8 , compares the A3C and EA3C algorithms, with A3C's results shown in Fig.8 (chart a) and EA3C's results in Fig.8 (chart b). The improvement in EA3C's performance is particularly evident in its handling of less frequent classes, which often pose challenges for classification models and algorithms due to data imbalance. In summary, the enhancement in EA3C's performance is attributed to a well-modified CNN architecture, which extracts high-level spatial and temporal features, improving feature representation compared to traditional fully connected A3C networks. Efficient parameter synchronization is achieved through optimized copy operations using sorted variable lists, ensuring precise weight transfers

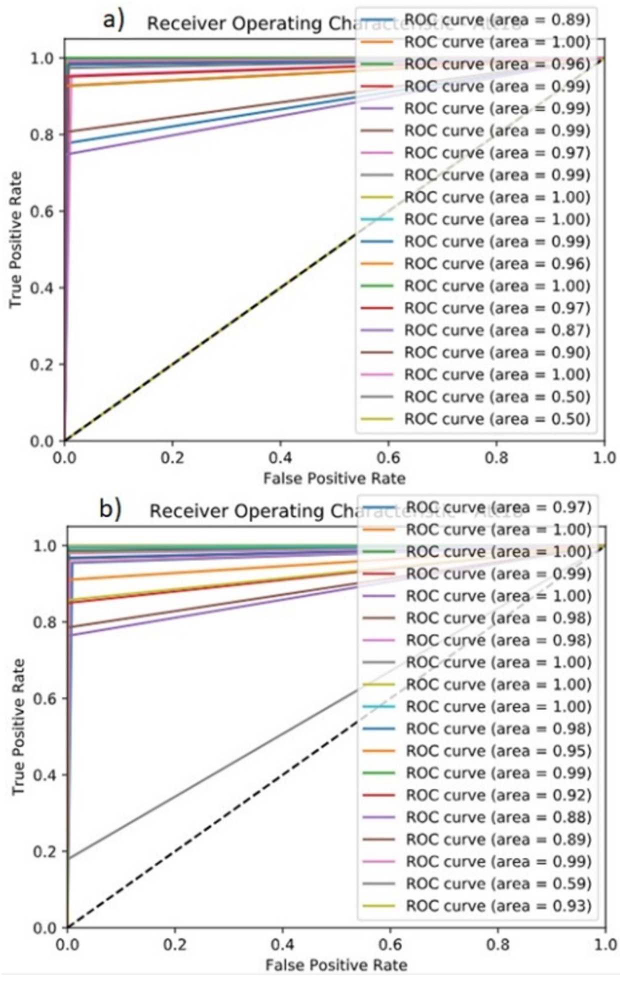


Figure 7. ROC Curve for X-IIoTID 19-Class a) Traditional A3C b) EA3C

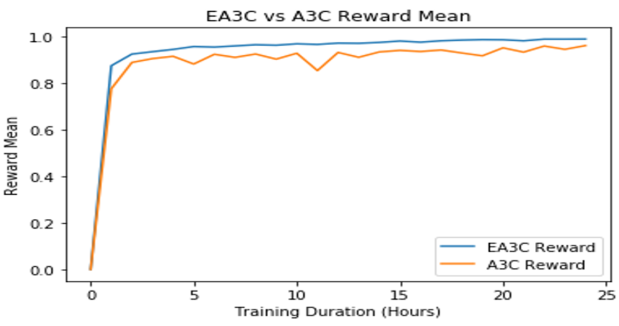


Figure 8. Reward Mean for X-IIoTID 19-Class for A3C and EA3C

between global and local networks, leading to faster convergence. Additionally, gradient clipping stabilizes training by controlling gradient updates, reducing variance in policy gradients, and preventing dominant classes from disproportionately influencing the algorithm. The input shaping is dynamically applied to all layers of the CNN, preventing mismatches in the data distribution across these layers. The use of

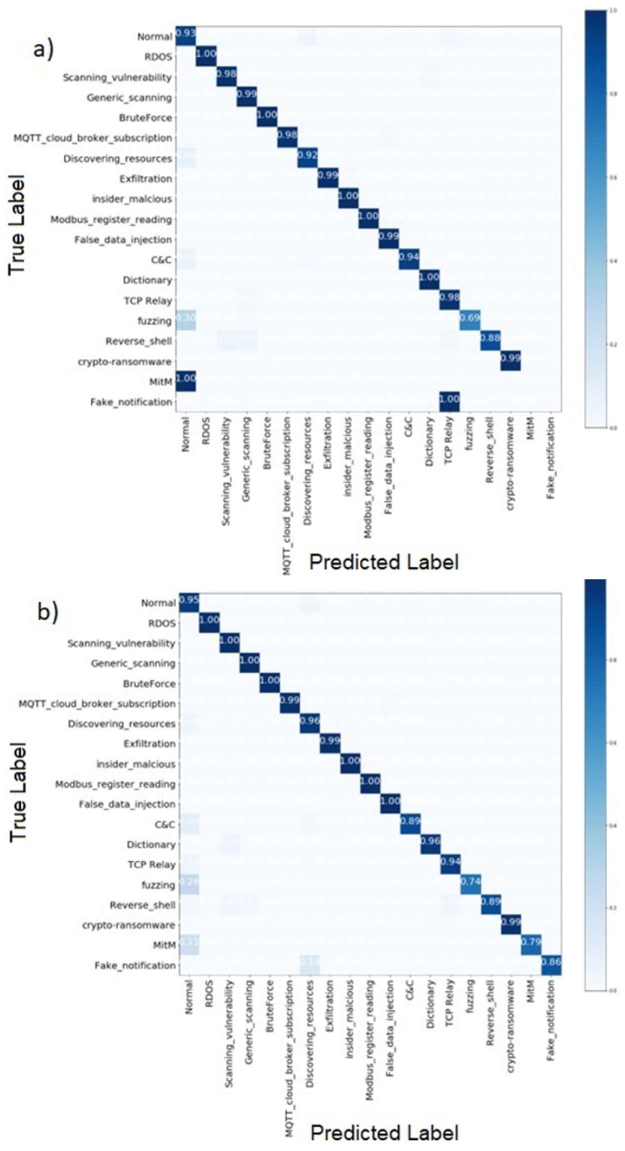


Figure 9. Enter Caption

policy and value estimations utilizes adaptive bootstrap rewards and advantage-based policy targets, enhancing policy generalization and accelerating convergence to enhance EA3C learning. By implementing parallel learning across different environments in EA3C, learning is accelerated across different environments while maintaining learning stability and recognizing different types of imbalanced cyberattacks. This makes this algorithm more effective for detecting intrusions in IIoT environments.

6. CONCLUSION AND FUTURE WORK

This research addresses the main challenges facing current IDSs in IIoT and edge computing networks. These challenges include poor intrusion detection performance with increasing classifications and the small number of class samples in the database, which negatively impacts the learning process. To overcome these challenges, the EA3C algorithm is proposed as an improvement over traditional A3C algorithms by replacing the traditional neural network with a modified CNN. This modified CNN consists of fully connected layers, convolutional layers used with dropout regularization to improve feature extraction, and an output layer is added to generate predictions for intrusion attack classifications. In addition, the optimized equations are used to calculate the effective synchronization parameter, gradient clipping to get stable training, and adaptive reward handling. This optimization significantly increases the algorithm's strength and efficiency in learning from imbalanced class sets.

The EA3C algorithm was implemented using the TensorFlow library and then tested using the X-IIoTID dataset, which is taken from a real IIoT environment. The results confirmed that the EA3C algorithm achieved strong weighted accuracy, precision, recall, and F1 scores metrics, all exceeding 0.98, compared to DT, AERL, DDQN, and traditional A3C. These results make EA3C a more efficient algorithm for improving network security.

Furthermore, these promising results indicate the potential for more development is required to improve EA3C performance in detecting zero-day attacks and expand its ability to handle real-world network traffic challenges. On the other side, it emphasizes the need to encourage researchers to explore and develop more applications of Actor-Critic algorithms in IDSs to enhance their robustness and scalability in dynamic IIoT environments.

REFERENCES

[1] T. Qiu, J. Chi, X. Zhou, *et al.*, "Edge computing in industrial internet of things: Architecture, advances and challenges," IEEE Commun. Surv. & Tutorials **22**, 2462–2488 (2020).

[2] B. Omoniwa, R. Hussain, M. A. Javed, *et al.*, "Fog/edge computing-based iot (feciot): Architecture, applications, and research issues," IEEE Internet Things J. **6**, 4118–4149 (2018).

[3] M. R. Asghar, Q. Hu, and S. Zeadally, "Cybersecurity in industrial control systems: Issues, technologies, and challenges," Comput. Networks **165**, 106946 (2019).

[4] M. A. Q. Al-Riyashi, A. T. Zahary, and F. Saeed, "A review for fog-cloud security: Aspects, attacks, solutions, and trends," Sana'a Univ. J. Appl. Sci. Technol. **2**, 482–491 (2024).

[5] E. Benkhelifa, T. Welsh, and W. Hamouda, "A critical review of practices and challenges in intrusion detection systems for iot: Toward universal and resilient systems," IEEE Commun. Surv. & Tutorials **20**, 3496–3509 (2018).

[6] N. M. Karie, N. M. Sahri, and P. Haskell-Dowland, "Iot threat detection advances, challenges and future directions," in *2020 Workshop on Emerging Technologies for Security in IoT (ETSecIoT),* (IEEE, 2020), pp. 22–29.

[7] A. M. Pasikhani, J. A. Clark, P. Gope, and A. Alshahrani, "Intrusion detection systems in rpl-based 6lowpan: A systematic literature review," IEEE Sensors J. **21**, 12940–12968 (2021).

[8] E. Muhati and D. B. Rawat, "Asynchronous advantage actor-critic (a3c) learning for cognitive network security," in *2021 Third IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA),* (IEEE, 2021), pp. 106–113.

[9] K. Zhou, W. Wang, T. Hu, and K. Deng, "Application of improved asynchronous advantage actor critic reinforcement learning model on anomaly detection," Entropy **23**, 274 (2021).

[10] H. Shen, K. Zhang, M. Hong, and T. Chen, "Towards understanding asynchronous advantage actor-critic: Convergence and linear speedup," IEEE Trans. on Signal Process. **71**, 2579–2594 (2023).

[11] G. Ma, Y. Bian, H. Qin, *et al.*, "Advance-fl: A3c-based adaptive asynchronous online federated learning for vehicular edge cloud computing networks," IEEE Trans. on Intell. Veh. (2024).

[12] A. Y. Abohatem, F. M. Ba-Alwi, and A. A. Al-Khulaidi, "Suggestion cybersecurity framework (csf) for reducing cyber-attacks on information systems," Sana'a Univ. J. Appl. Sci. Technol. **1** (2023).

[13] F. Hussain, S. G. Abbas, I. M. Pires, *et al.*, "A two-fold machine learning approach to prevent and detect iot botnet attacks," IEEE Access **9**, 163412–163430 (2021).

[14] N. Moustafa, B. Turnbull, and K.-K. R. Choo, "Towards automation of vulnerability and exploitation identification in iiot networks," in *2018 IEEE International Conference on Industrial Internet (ICII),* (IEEE, 2018), pp. 139–145.

[15] C. Avasalcai and S. Dustdar, "Edge computing: Use cases and research challenges," in *Digital Transformation: Core Technologies and Emerging Topics from a Computer Science Perspective,* (2023), pp. 125–142.

[16] M. Eceiza, J. L. Flores, and M. Iturbe, "Fuzzing the internet of things: A review on the techniques and challenges for efficient vulnerability discovery in embedded systems," IEEE Internet Things J. **8**, 10390–10411 (2021).

[17] F. Maulana, H. Fajri, M. F. Safitra, and M. Lubis, "Unmasking log4j's vulnerability: Protecting systems against exploitation through ethical hacking and cyberlaw perspectives," in *2023 9th International Conference on Computer and Communication Engineering (ICCCE),* (IEEE, 2023), pp. 311–316.

[18] S. Salamatian, W. Huleihel, A. Beirami, *et al.*, "Why botnets work: Distributed brute-force attacks need no synchronization," IEEE Trans. on Inf. Forensics Secur. **14**, 2288–2299 (2019).

[19] I. Alkhwaja, M. Albugami, A. Alkhwaja, *et al.*, "Password cracking with brute force algorithm and dictionary attack using parallel programming," Appl. Sci. **13**, 5979 (2023).

[20] M. Omar, D. Mohammed, and V. Nguyen, "Defending against malicious insiders: a conceptual framework for predicting, detecting, and deterring malicious insiders," Int. J. Bus. Process. Integr. Manag. **8**, 114–119 (2017).

[21] M. Al-Hawawreh and E. Sitnikova, "Developing a security testbed for industrial internet of things," IEEE Internet Things J. **8**, 5558–5573 (2020).

[22] S. Gönen, M. A. Barişkan, D. Y. Kaplan, *et al.*, "A novel approach detection for false data injection, and man in the middle attacks in iot and iiot," in *2023 IEEE PES GTD International*

Conference and Exposition (GTD), (IEEE, 2023), pp. 278–282.
[23] J. Bi, S. He, F. Luo, *et al.*, "Defense of advanced persistent threat on industrial internet of things with lateral movement modelling," IEEE Trans. on Ind. Informatics (2022).
[24] R. Gupta, N. K. Jadav, H. Mankodiya, *et al.*, "Blockchain and onion-routing-based secure message exchange system for edge-enabled iiot," IEEE Trans. on Ind. Informatics **19**, 1965–1976 (2022).
[25] S. N. Firdous, Z. Baig, C. Valli, and A. Ibrahim, "Modelling and evaluation of malicious attacks against the iot mqtt protocol," in *2017 IEEE International Conference on Internet of Things (iThings), GreenCom, CPSCom, and SmartData,* (IEEE, 2017), pp. 748–755.
[26] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "Asynchronous peer-to-peer federated capability-based targeted ransomware detection model for industrial iot," IEEE Access **9**, 148738–148755 (2021).
[27] M. A. Ferrag, O. Friha, D. Hamouda, *et al.*, "Edge-iiotset: A new comprehensive realistic cyber security dataset of iot and iiot applications for centralized and federated learning," IEEE Access **10**, 40281–40306 (2022).
[28] D. Tychalas, A. Keliris, and M. Maniatakos, "Led alert: Supply chain threats for stealthy data exfiltration in industrial control systems," in *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS),* (IEEE, 2019), pp. 194–199.
[29] M. Al-Hawawreh, F. D. Hartog, and E. Sitnikova, "Targeted ransomware: A new cyber threat to edge system of brownfield industrial internet of things," IEEE Internet Things J. **6**, 7137–7151 (2019).
[30] H. Mankodiya, N. K. Jadav, S. Tanwar, and R. Gupta, "Deep learning-based secure machine-to-machine communication in edge-enabled industrial iot," in *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS),* (IEEE, 2022), pp. 48–53.
[31] F. Thabit, S. Alhomdy, and S. Jagtap, "A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions," Int. J. Intell. Networks **2**, 18–33 (2021).
[32] Y. Wu, H.-N. Dai, and H. Wang, "Convergence of blockchain and edge computing for secure and scalable iiot critical infrastructures in industry 4.0," IEEE Internet Things J. **8**, 2300–2317 (2020).
[33] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation ethernet, iiot, and 5g," Proc. IEEE **107**, 944–961 (2019).
[34] S. Bhattacharjee, *Practical Industrial Internet of Things security: A practitioner's guide to securing connected industries* (Packt Publishing Ltd, 2018).
[35] Z. Liu, Q. Liu, L. Wang, *et al.*, "Task-level decision-making for dynamic and stochastic human-robot collaboration based on dual agents deep reinforcement learning," The Int. J. Adv. Manuf. Technol. **115**, 3533–3552 (2021).
[36] M. Natarajan and A. Kolobov, *Planning with Markov decision processes: An AI perspective* (Springer Nature, 2022).
[37] W. E. L. Ilboudo, T. Kobayashi, and K. Sugimoto, "Robust stochastic gradient descent with student-t distribution based first-order momentum," IEEE Trans. on Neural Networks Learn. Syst. **33**, 1324–1337 (2020).
[38] S. Ravichandiran, *Hands-on meta learning with Python: meta learning using one-shot learning, MAML, Reptile, and Meta-SGD with TensorFlow* (Packt Publishing Ltd, 2018).
[39] N. D. Nguyen, T. T. Nguyen, P. Vamplew, *et al.*, "A prioritized objective actor-critic method for deep reinforcement learning," Neural Comput. Appl. **33**, 10335–10349 (2021).
[40] S. Demir, B. Stappers, K. Kok, and N. G. Paterakis, "Statistical arbitrage trading on the intraday market using the asynchronous advantage actor–critic method," Appl. Energy **314**, 118912 (2022).
[41] P. Jayalaxmi, R. Saha, G. Kumar, *et al.*, "Pignus: A deep learning model for ids in industrial internet-of-things," Comput. & Secur. p. 103315 (2023).
[42] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Trans. on Emerg. Top. Comput. Intell. **2**, 41–50 (2018).
[43] M. Al-Hawawreh, E. Sitnikova, and N. Aboutorab, "X-iiotid: A connectivity-agnostic and device-agnostic intrusion data set for industrial internet of things," IEEE Internet Things J. **9**, 3962–3977 (2021).
[44] R. Vinayakumar, M. Alazab, K. Soman, *et al.*, "Deep learning approach for intelligent intrusion detection system," IEEE Access **7**, 41525–41550 (2019).
[45] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," IEEE Access **5**, 21954–21961 (2017).
[46] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," IEEE Access **7**, 30373–30385 (2019).
[47] S. Tharewal, M. W. Ashfaque, S. S. Banu, *et al.*, "Intrusion detection system for industrial internet of things based on deep reinforcement learning," Wirel. Commun. Mob. Comput. **2022**, 1–8 (2022).
[48] H. Alavizadeh, H. Alavizadeh, and J. Jang-Jaccard, "Deep q-learning based reinforcement learning approach for network intrusion detection," Computers **11**, 41 (2022).
[49] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," Comput. Networks **159**, 96–109 (2019).
[50] A. Zafar, M. Aamir, N. M. Nawi, *et al.*, "A comparison of pooling methods for convolutional neural networks," Appl. Sci. **12**, 8643 (2022).
[51] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning,* (PMLR, 2016), pp. 1928–1937.
[52] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "Optimality and approximation with policy gradient methods in markov decision processes," in *Conference on Learning Theory,* (PMLR, 2020), pp. 64–66.
[53] L. Li, D. Li, T. Song, and X. Xu, "Actor–critic learning control with regularization and feature selection in policy gradient estimation," IEEE Trans. on Neural Networks Learn. Syst. **32**, 1217–1227 (2020).