Edelweiss Applied Science and Technology
ISSN: 2576-8484
Vol. 9, No. 11, 1053-1075
2025
Publisher: Learning Gate
DOI: 10.55214/2576-8484.v9i11.11054
© 2025 by the authors; licensee Learning Gate

Hybrid IIoT intrusion detection enhancement with A3C

Abdulssalam M Khako[1*], Sharaf A Alhomdy[2]
[1,2]Department of Information Technology, Faculty of Computer and Information Technology, Sana'a University, Sana'a, Sanaa, Yemen; abs-khako@hotmail.com (A.M.K.) sha.alhomdy@su.edu.ye (S.A.A.).

Abstract: Traditional Intrusion Detection Systems often face difficulties in detecting the diverse classes of intrusions in the Industrial Internet of Things (IIoT) environment due to its heterogeneity and complexity. Such environments encounter a mixture of deterministic and stochastic cyberattack patterns, corresponding to discrete and continuous features. This challenge results in inefficient feature extraction in current Asynchronous Advantage Actor-Critic (A3C) algorithms. To address these issues, this article proposes a Hybrid A3C with Decision Tree (HADT) algorithm for intrusion detection. In this proposed algorithm, the simple A3C is used to detect stochastic intrusions related to continuous features, while the Decision Tree (DT) algorithm is employed for detecting fixed-pattern (deterministic) intrusions related to discrete features. The outputs of these algorithms are combined in a Feedforward Neural Network (FFNN) layer to accurately classify the intrusion type. Additionally, the X-IIoTID dataset, obtained from a real IIoT environment, was used to compare the performance of HADT with existing A3C algorithms in related studies. Experimental results demonstrate that the HADT algorithm outperforms baseline A3C algorithms in terms of accuracy, precision, recall, and F1 score, maintaining a detection rate above 99.8%, especially for intrusions that are underrepresented in the dataset, while also consuming less power.

Keywords: *A3C, DT, FFNN, HADT, IIoT, X-IIoTID.*

1. Introduction

In the IIoT environment, making informed decisions is crucial when exchanging data efficiently across multiple domains. However, the rapid growth of this environment has created significant challenges in cybersecurity, requiring sophisticated intrusion detection systems (IDS) [1]. Although basic protection is provided for the IIoT network, current security technologies, such as firewalls and anti-theft mechanisms, are still insufficient to address these cyber challenges [2]. These technologies often fail to detect skilled hackers who use complex and sophisticated intrusions [3]. These intrusions could lead to huge financial and economic losses.

Current IDSs used for intensive and anomaly detection have faced several performance measurement problems, including limited detection accuracy, precision, recall, and F1 score [4]. These problems reduce the effectiveness of securing IIoT networks [5]. In recent years, actor-critic algorithms have emerged to address these challenges. They have been developed with adaptive learning capabilities to improve IDSs [6].

Actor-critic algorithms utilize neural network (NN) layers within workers. However, these algorithms face computational challenges when processing large-scale traffic data or high-volume network activities [7]. Furthermore, the intrusion detection capability of the IIoT environment, which is a mixture of stochastic and deterministic environments, remains limited. This poses additional obstacles during the training and testing phases [8]. To overcome these challenges, researchers have proposed various enhancements. For instance, the A3C algorithm was proposed to use CNN inside its workers for anomaly detection [9]. Similarly, in optimized A3C, policies and value function parameters were

History: Received: 2 September 2025; Revised: 10 October 2025; Accepted: 14 October 2025; Published: 17 November 2025
* Correspondence: abs-khako@hotmail.com

proposed to enhance security in cognitive networks [10]. Another study, Lim and Joe [11], the authors explored off-policy actor-critic DRL techniques for prioritizing alerts in IDS environments by describing the effectiveness of DRL in complex security scenarios.

Despite these improvements, there must be a balance between exploration and exploitation trade-offs in IDSs based on actor-critic algorithms such as A3C, which remains a major concern, especially in complex environments such as IIoT environments that have continuous and discrete intrusion patterns [12]. To address this, the article proposes an HADT algorithm, which is a hybrid integrating an A3C algorithm with a DT algorithm. This integration aims to mitigate the exploration and exploitation problem while improving classification performance. Using HADT algorithms for stochastic and deterministic environments is more effective in dealing with intrusion patterns that rely on continuous and discrete features, respectively, with fewer neural network layers to reduce processing operations, which contributes to improving processing time and power consumption.

On the other hand, to benefit from the effectiveness of the proposed algorithm, it requires pre-processing the dataset before the training process to ensure the efficiency of feature extraction. The results of the HADT algorithm confirm enhanced intrusion detection accuracy, precision, recall, and F1-score, which make it a promising solution for securing complex IIoT networks from modern cyber intrusions.

1.1. Problem Statement

The IIoT technologies have been developed to enable seamless connectivity and automation in industries. However, this advancement has also introduced modern cyber threats, exposing IIoT networks to intrusions that are most relevant to continuous features like Distributed Denial of Service (DDoS) and reconnaissance [13] as well as intrusions that are most relevant to discrete features such as data theft and malware exploits [14]. Since the A3C can be used with discrete features, it suffers from a lack of smoothness and poor generalization because this algorithm provides smooth numeric input to stabilize learning. On the other hand, using the DT algorithm is more suitable for these intrusions. Although the DT is also used to deal with continuous features in a dataset, it requires specifying the threshold in advance to split features at certain values. Inaccurate threshold specification reduces attack classification performance.

1.2. Motivation

This article focuses on increasing the efficiency of intrusion detection in the IIoT network against hackers, as traditional IDSs cannot handle these threats effectively due to the increasing complexity of the expansion and the architecture of IIoT networks in industrial fields. Hence, the intrusion detection mechanism must be robust and adaptable. Therefore, the use of HADT can overcome the shortcomings of traditional IDSs through policy-based learning and improved decision rules. By utilizing the efficient hybrid algorithm and the X-IIoTID dataset, this article seeks to create a high-quality IDS that can enhance intrusion detection accuracy, precision, F1-score, and recall in IIoT environments.

1.3. Contribution

This article proposes the HADT algorithm with fewer neural network layers, which integrates a simple A3C with decision trees. This integration addresses the shortcomings of traditional A3C for intrusion detection systems (IDSs). The article combines different complementary machine learning techniques and employs a contemporary dataset to develop specialized approaches for Industrial Internet of Things (IIoT) networks. The main highlights of this article are:

- Using the X-IIoTID dataset to address real-world IIoT environments, which contains two types of features continuous and discrete representing a general and realistic view of data traffic in the IIoT network.

- Hybridizing simple A3C with DT using fewer NN layers helps reduce processing operations, thereby improving processing time and power consumption. This approach integrates the advantages of A3C with DT into another FFNN layer, which helps to get the advantages of both algorithms in detecting intrusions.
- Combining improved policy learning with decision rules significantly increases the efficiency of intrusion detection under different intrusions.

This is shown through iterative testing and evaluation of the proposed algorithm using accuracy, precision, recall, and F1-score metrics, and by comparing the results with those of existing A3C algorithms. Through these contributions, the proposed algorithm ensures effective identification and detection of IIoT intrusions.

2. Related Work

With the rapid development of IIoT networks, actor-critic algorithms have been developed to support robust IDSs, especially amid the emergence of new intrusion patterns that are increasingly difficult to detect. Being one of the actor-critic algorithms, A3C contributes significantly to current intrusion detection. However, its traditional forms have become ineffective due to the complexity of IIoT environments that incorporate both stochastic and deterministic intrusion patterns in the IIoT network [15]. A study by Muhati and Rawat [16] suggested an A3C-based cognitive network intrusion detection system (NIDS) using deep reinforcement learning (DRL) to adaptively analyze network traffic. Utilizing open-source service discovery tools, it has improved intrusion detection in network attacks, achieving high accuracy on various datasets, including UNSW-NB15, AWID, and NSL-KDD. Similarly, Zhou et al. [17] examined an improved A3C algorithm for intrusion detection, incorporating attention mechanisms and CNN networks. This improvement demonstrates that RL surpasses traditional machine learning methods in intrusion detection, with improved accuracy and recall among participants as well as F1 score for high precision across public benchmark datasets (NSL-KDD, AWID CICIDS-2017, and DoHBrw-2020).

Integrating hybrid models into RL-based IDSs enhances the system's ability to adapt to both stochastic and deterministic network environments. Shen et al. [18] suggested a Mean-Field Game-based Asynchronous Advantage Actor-Critic (MFGA3C) model to counter malware propagation in Edge Intelligence-based IoT (EIoT) networks. Their model improves multi-worker reinforcement learning (MARL) to advance optimal defense policies against adversarial attacks, demonstrating improved resilience in malware mitigation.

A dynamic threshold mechanism is used by the Bayesian DRL framework of Watts et al. [9], who developed specialized techniques for real-time intrusion classification. They combine a CNN with essentially any partially observable Markov decision process (POMDP), which allows for very robust intrusion detection in automated systems.

To improve detection accuracy, effective IDS frameworks of A3C algorithms require extensive datasets to train and validate their algorithms. As stated by Dong et al. [19], they demonstrated that feature selection and data preprocessing are crucial factors for systematically refining high-dimensional network traffic data in the optimization of deep RL-based IDSs.

In a previous research article presented by the authors Anbazhagan and Mugelan [20], the Enhanced A3C (EA3C) algorithm was compared with several recent algorithms such as DT, Attention-Enhanced Reinforcement Learning (AERL), Double Deep Q-Network (DDQN), and traditional A3C. The experiments indicated that the EA3C algorithm outperformed these algorithms by 1.3–1.7% for accuracy, precision, recall, and F1-score, particularly on handling low-sample-size attacks and mixed feature-type environments, as empirically verified on the X-IIoTID dataset. All of these improvements are due to the improved CNN architecture of the EA3C worker, the training stability achieved by gradient clipping, and the improved parameter synchronization mechanism. However, it suffers from the fact that it requires a long time to train, in addition to the fact that it requires complex calculations, which causes a large consumption of power.

This leads to proposing the HADT algorithm, which enhances detection performance through effective training of IIoT network data, especially in low-sample-size attacks. Moreover, the use of the X-IIoTID dataset in the HADT algorithm provides an opportunity to overcome existing limitations in IDS datasets. Unlike previous studies that relied heavily on specific datasets such as NSL-KDD, AWID, and CICIDS-2017, this study uses the X-IIoTID dataset, allowing for testing algorithms in real IIoT environments. Therefore, the HADT algorithm, integrating simple A3C and DT algorithms, was proposed to handle intrusions in both continuous and discrete features and to improve the detection of various classes of intrusions. By utilizing advanced dataset preprocessing strategies and sophisticated algorithmic approaches, the proposed work facilitates the development of effective and adaptive IDS solutions for IIoT security. In Table 1, a thorough comparison of A3C-based network intrusion detection methods is presented, emphasizing their study algorithms, NN architectures, datasets, strengths, and limitations. These related works are summarized below:

Table 1.
Related work summary.

Ref.	**Publisher**	**Study Methods**	**Type of NN in Worker**	**Dataset**	**Strength of Methods**	**Limitations of Methods**
Muhati and Rawat [16]	Muhati and Rawat [16]	A3C-based cognitive NIDS with DRL	FFNN	UNSW-NB15, AWID, NSL-KDD	Adaptive to evolving attack patterns, improving anomaly detection accuracy.	Limited adaptability to new attack patterns due to static feature selection.
Zhou et al. [17]	Zhou et al. [17]	A3C algorithm with attention & CNN	CNN + Attention	NSL-KDD, AWID, CICIDS-2017, DoHBrw-2020	High precision, recall, and F1-score for anomaly detection	High computational costs limit real-time deployment in IIoT.
Shen et al. [18]	Shen et al. [18]	Mean-Field Game-based A3C (MFGA3C)	FFNN	Not specified	Effective against adversarial attacks, enhancing malware mitigation.	Scalability issues in large IIoT networks with high traffic loads
Watts et al. [9]	Watts et al. [9]	Bayesian DRL with CNN & POMDP	CNN + POMDP	Not specified	Robust real-time anomaly detection with adjustable thresholds.	Requires extensive training data; struggles with low sample size attacks in the dataset.
Dong et al. [19]	Dong et al. [19]	Feature selection & DRL-based IDS	FFNN	Not specified	Optimizing detection accuracy through feature selection.	High-dimensional feature space increases training complexity.
Khako and Alhomdy [21]	Khako and Alhomdy [21]	EA3C with gradient clipping, and an improved parameter synchronization mechanism	CNN	X-IIoTID	Handling low-sample-size attacks and mixed feature-type environments	It requires a long time to train and complex calculations.

3. Proposed HADT

The proposed HADT is a combination of two algorithms, which are the simple A3C and DT. In a continuous feature, A3C can be used to efficiently calculate policy functions and predict intrusion actions. Adding a DT unit was necessary to address features of the deterministic environment and improve intrusion detection performance by selecting optimal actions for detecting attacks. To achieve this, an

FFNN layer is proposed to hybridize the A3C-FFNN and DT outputs in the HADT algorithm, as illustrated in Figure 1.

As described, the HADT algorithm integrates multiple learning algorithms to enhance intrusion detection in IIoT environments as follows:

1) A3C-FFNN whose worker has a simple FFNN layer that produces initial policy and value functions. It consists of a two-layer neural network connected to an input layer, followed by a dense layer with ReLU activation and a dropout layer for regularization. The algorithm has two output layers: one for the policy function activated with a softmax function, and the value function produced with a linear function. The FFNN layer in A3C-FFNN is compiled using the Adam optimizer.

2) The DT algorithm is used to make action predictions in parallel with the A3C-FFNN. This algorithm has automatic feature selection and interactions and the capability to offer explainable decision rules, leading to reduced overfitting, increased accuracy, and more reliable estimates of probability for intrusion classification.

3) The FFNN layer hybridizes the predictions of both the DT and the A3C-FFNN as inputs. Concatenation of policy predictions is achieved through a hidden layer that is activated with ReLU, and regularization is accomplished through the use of a dropout layer. Compiling the output policy function with the Adam optimizer and sparse categorical cross-entropy loss is necessary for the hybrid FFNN output layer, which uses the softmax activation function. The convergence effect of the network is maintained by adjusting its weights and biases repeatedly, which requires frequent training. This layer is added to leverage the strengths of both the A3C-FFNN and DT algorithms, potentially leading to a generalized improvement in intrusion detection prediction performance due to its ability to capture complex interactions between dataset features. The HADT algorithm inherently uses features based on their statistical characteristics. They are separated by correlation analysis and skewness testing and are fed to A3C-FFNN for the continuously distributed and high-variance features and to DT for the discrete or categorically distributed features. This separation is realized in Algorithm 1 to realize maximum feature-type allocation for enhanced detection accuracy.

Algorithm 1: Automatic Feature-Type Separation for HADT

```
Input: Feature set ( F ), threshold ( $\tau = 0.5$ )
Output: Continuous features ( F_c ), Discrete features ( F_d )
1. for each feature ( f_i ) in ( F ):
2.        Compute skewness ( s_i ) and correlation ( c_i )
3..       if ( $|s_i| < \tau$ ) and ( $c_i > \tau$ ):
```

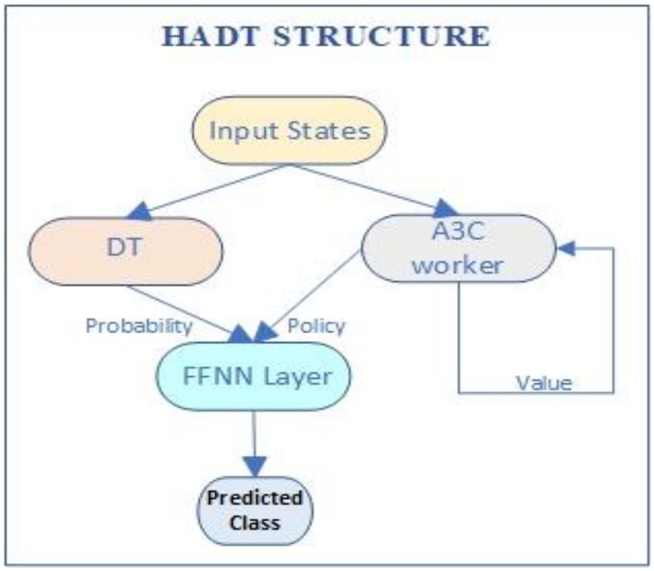

Figure 1.
The structure of the HADT algorithm.

The HADT algorithm employs a limited number of neural network layers to strike a balance between detection performance and computational efficiency. This design choice is based on empirical analysis, which showed that increasing layers beyond this point yielded diminishing returns in accuracy while significantly raising power consumption and inference time. Thus, the selected architecture is optimal for real-time IIoT intrusion detection at the edge.

The detailed structure of the HADT algorithm is shown in Figure 2, which helps to improve the results of the predicted intrusion class 'actions' and to obtain the best accuracy.

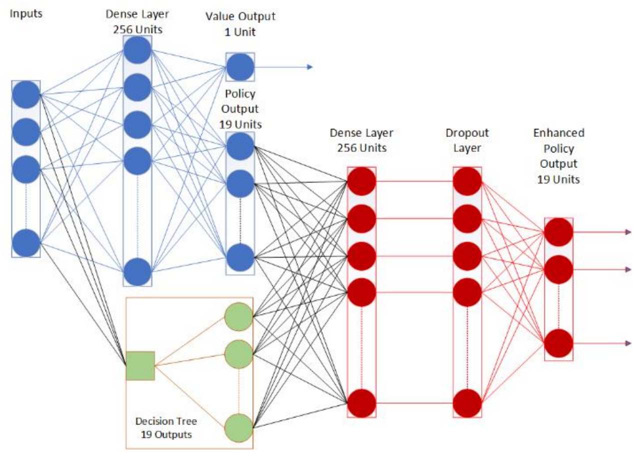

Figure 2.
The layers of the Proposed HADT algorithm.

The actor in A3C-FFNN seeks to predict a policy $\pi(s)$ by getting criticism from an advantage function. The expected return in rewards, or value function V(s), is taught to the critic and is used to calculate the relative advantages [22]. If s is the current state, s' is the next state, r is the current reward, γ is the discount factor, and $V(s)$ is the value function, then the advantage is defined as $A(s) = r+\gamma V(s') - V(s)$. The expected value or reward associated with a specific condition is predicted using the value function. A linear activation function is employed to obtain the expected value, as shown in Equation (1).

$$V(s;\theta) = W \cdot \varphi(s) \tag{1}$$

Where W is the weight matrix connecting the hidden layer to the value output, and V is the predicted value of states s given the FFNN layer's parameters θ. Given a state, the policy function predicts the probability distribution across possible actions. Ordinarily, a softmax activation function is used in Equation (2) to calculate the probabilities of actions.

$$\pi(a \mid s;\theta) = softmax(W \cdot \varphi(s)) \tag{2}$$

Here, $\pi(a \mid s;\theta)$ is the probability of taking action a and $\varphi(s)$ is the hidden layer's features, or output for state s. In Equation (3), the loss function is formulated based on the difference between the actual and predicted values [23]. Specifically, R_i represents the actual return, which is computed from the observed rewards in the IIoT environment. The predicted state-value function, denoted as $V(s_i;\theta)$, estimates the expected return given s_i and θ. The loss is computed over a batch of N samples, where N represents the number of s used in each training iteration. This formulation ensures that the FFNN layer in A3C-FFNN effectively learns to minimize the divergence between predicted and actual returns, enhancing the accuracy of the RL process.

$$L_v = \frac{1}{N}\sum_{i=1}^{N} \left(R_i - V(s_i;\theta)\right)^2 \tag{3}$$

From Equation (4), the policy loss function is a key component in policy gradient methods. This equation measures how well the policy π selects actions that lead to higher rewards [24]. The expectation operator E represents the average over multiple sampled states and actions. The term A(s, a) is the advantage function, which indicates how much better or worse a specific action a is compared to the average a at s. The logarithm of the policy function ensures that the FFNN layer in the A3C worker updates in proportion to how confident it is in selecting a. The negative sign in the function is necessary because optimization algorithms typically perform gradient descent, which minimizes loss functions. By minimizing this loss, the algorithm maximizes the probability of choosing advantageous actions while reducing the probability of selecting suboptimal ones, leading to an improved policy over time.

$$L_p = -E(A(s;\theta) log(\pi(a \mid s))) \tag{4}$$

Thus, it is crucial to include the entropy $H(\pi)$ in the loss function equations as expressed in Equation (5).

$$H(\pi) = -\Sigma\left[\pi(a \mid s;\theta)\ log(\pi(a \mid s;\theta))\right] \tag{5}$$

A measure of the dispersion of probability is called entropy. The higher the entropy, the more similar the likelihood of any a and the more uncertain the worker will be about which course of a to take [20]. Adding entropy to the loss function, as expressed in Equation (6), will then encourage exploration by preventing the worker from being very definite and converging to a local optimum.

$$L_p = -E(A(s;\theta) log(\pi(a \mid s))) - \beta * H(\pi) \tag{6}$$

The loss function for the entire layer is obtained by combining the two loss functions as in Equation (7)

$$L_\theta = \mathrm{L}_V + L_p \tag{7}$$

Ensemble usage of the DT technique will output class probabilities. These probabilities are computed as in Equation (8) from the percentage of samples within the leaf node that fall into a specific class.

$$P(a \mid s) = \frac{n_a}{n} \tag{8}$$

Where n_a is the number of samples associated with action 'a' in the leaf node, n is the total number of samples in the leaf node, and P is the probability of executing a given state s. Final probabilities are then obtained by averaging all the predictions made by the DTs in the ensemble. Each DT predicts the probability of a class independently. This process of averaging gives robustness and stability to the predictions.

The DT algorithm is based on the Gini impurity, which serves as a loss function to measure the impurity of a node, as shown in Equation (9), estimating the likelihood that a randomly selected state from the dataset in a node would be incorrectly classified.

$$L_G(p) = 1 - \sum (p_i^2) \tag{9}$$

Where L_G is the Gini impurity of a node and P_i is the proportion of samples in that node that belong to class i. The training predictions from the A3C-FFNN algorithm and the DT algorithm are combined in the FFNN layer. The outputs of the predictions from the A3C-FFNN and the DT algorithms define the input layers of the FFNN layer. The following Equation (10) represents the results of the FFNN layer F from concatenating these predictions, represented by $\pi(a \mid s;\theta)$ with $P(a \mid s)$.

$$F(a \mid s) = \pi(a \mid s;\theta) \square P(a \mid s) \tag{10}$$

So, the total loss function from FFNN and DT algorithms will be calculated as in Equation (11):

$$L_{p'} = 0.5 * \Sigma\big(R - V(s;\theta)\big)^2 - log\big(\pi(a \mid s)\big) * A(s;\theta) - \beta * H(\pi) + L_G(p) \tag{11}$$

As training, the L_{TOTAL} total loss function is a combination of the loss from the A3C-FFNN and DT algorithms. The compound loss is optimized using the Adam optimizer, which automatically updates the layer weights via backpropagation. The HADT algorithm optimizes the policy, value, and decision tree components simultaneously with sustained generalization, detection accuracy, and convergence over various intrusion patterns.

On the other hand, a hidden layer with 256 units and ReLU activation, followed by dropout regularization, is applied to reduce overfitting. The output layer then provides class probabilities using the softmax activation function. Therefore, the improved police output, in terms of the police output before improvement, will be as in Equation (12):

$$Q(a \mid s) = \frac{e^{(\max(0,\pi(a|s;\theta)\square P(a|s))*m).w+b)_i}}{\sum_{j=1}^{n} e^{(\max(0,\pi(a|s;\theta)\square P(a|s))*m).w+b)_j}} \tag{12}$$

With the Adam optimizer, a batch size of 512, and the sparse categorical cross-entropy loss function, the resulting equation builds an FFNN layer across training iterations for up to 50 epochs with early stopping, as shown in Algorithm 2, if the validation loss decays with a learning rate of 0.001. The A3C-FFNN and DT algorithms' predictions use validation data to assess an algorithm's performance during the training of the FFNN layer.

Algorithm2: Hybrid HADT algorithm

```
1   //Compute A3C and DT probabilities.
2   $P_{DT}$:= DT_predict_proba(X′)
3   $P_{A3C}$:= A3C_predict_proba (X′)
4   for e=1 to 50 do                // Loop for training over epochs.
```

```
    $F := P_{DT,} || P_{A3C})$      //Concatenate features from both Algorithms
    $H := \sigma(W_1F+b_1)$      //Apply fully connected layer.
    $H' := Dropout(H,p=0.5)$    //Apply dropout for regularization.
    $Q := softmax(W2H'+b2)$      //Compute output layer.
    $L := -\sum y\log(Q)$  // loss function (Sparse Categorical Cross-Entropy).
    $\theta \leftarrow \theta - \alpha\nabla L$      //Update layer weights using Adam optimizer.
    $M(e) := train(PDT,PA3C,y,512)$   // Output trained hybrid layer
    //Early stopping if  validation loss does not improve for 5
    if  $L_{val}(e) \geq L_{val}(e-5)$ then break else continue
end for
$M_{final} := Q$                    //Output trained hybrid layer.
```

4. Proposed Methodology

The article methodology relies on comprehensive pre-processing of the X-IIoTID dataset and training layers of the HADT algorithm, which achieves performance improvements, including accuracy, precision, recall, and F1 score. The phases of this methodology, as described in Figure 3, are an effective tool for identifying different imbalanced classes of attacks that belong to a mixed environment, which is stochastic or deterministic. Finally, the testing results of the HADT algorithm are evaluated by comparing them with existing A3C algorithms.

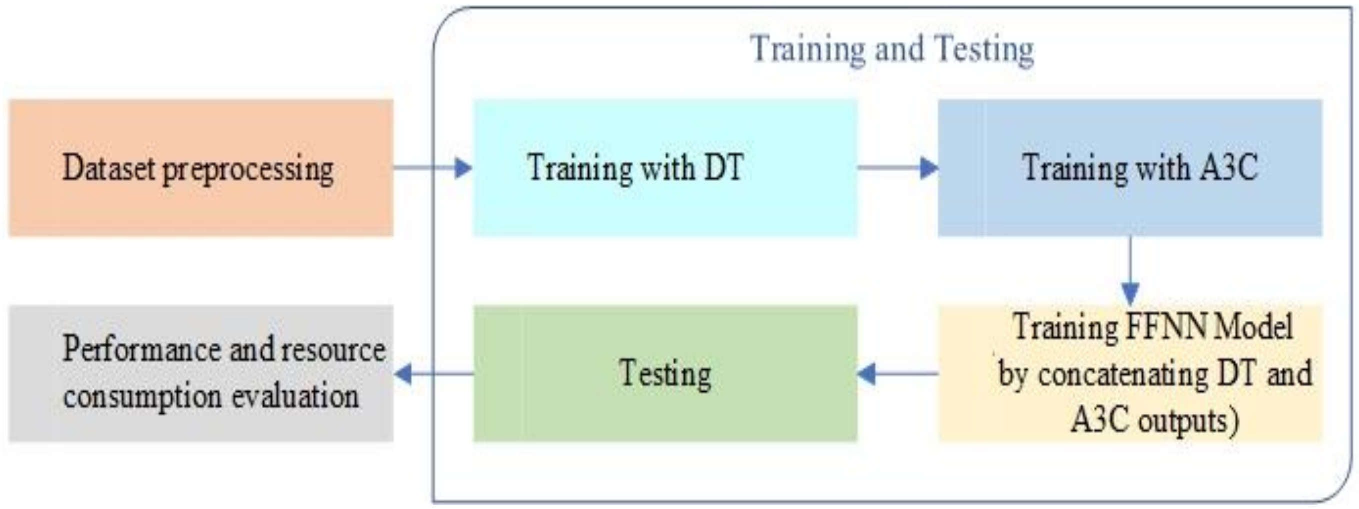
Training and Testing
Dataset preprocessing
Training with DT
Training with A3C
Training FFNN Model by concatenating DT and A3C outputs)
Testing
Performance and resource consumption evaluation

Figure 3.
Methodology phases.

4.1. Phase 1. Dataset Preprocessing

In this phase, preprocessing is essential to prepare the X-IIoTID dataset for use in training and testing processes for intrusion attack classes. Each of these attacks has different data traffic features. This phase consists of three steps as follows:

-Step 1: Class Label Encoding: The dataset was split twice. The first time, it was divided into two main categories: normal traffic and malicious data. This is called binary classification. The second time, the classification was based on normal traffic and 18 intrusion data classes. Normal traffic represents legitimate network activity. RDOS (Resource Depletion Denial-of-Service) and brute-force attacks aim to overwhelm or gain unauthorized access to systems. Scanning vulnerability and generic scanning involve probing networks for weaknesses. MQTT cloud broker subscription and resource discovery target IoT communication protocols and resource identification. Exfiltration refers to unauthorized data extraction, while insider malicious threats originate from within the organization. Modbus register reading and false data injection can manipulate industrial control systems. C&C (Command and Control) attacks enable remote control of compromised devices. Dictionary and TCP Relay attacks focus on credential-based intrusions and traffic redirection. Fuzzing tests system vulnerabilities by injecting malformed data. A reverse shell provides attackers with remote system access. Crypto-ransomware encrypts critical data for ransom. Man-in-the-Middle (MitM) attacks intercept communication, while Fake notifications deceive users with fraudulent alerts. These attack categories highlight the diverse threats facing IIoT networks and emphasize the need for robust Intrusion Detection Systems (IDSs).

-Step 2: Feature Encoding: In this step, categorical features in the dataset, such as service and protocol, are converted to numeric values. IPv4 and IPv6 addresses are also converted to numeric values to facilitate the processing and analysis of network traffic data.

In addition, missing values are replaced with -1, while NaN values are pre-filled to prevent data inconsistencies during training and testing. By using uniform processing, feature values are bounded between [0, 1] to simplify calculations and remove dimensionality, as shown in Equation (13).

$$V_N = \frac{V - V_{\min}}{V_{\max} - V_{\min}} \tag{13}$$

Where V_{max} and V_{min} represent the highest and lowest feature values, respectively.

Step 3: Train-Test Split: The dataset is divided into a training set (70%), of which 20% is allocated for validation during the training process, and a test set (30%), as shown in Table 2, to determine the algorithm's performance when using the X-IIoTID dataset. Learning is therefore performed on the training set. The algorithm is evaluated on a separate test set to determine its detection ability.

Table 2.
Train-Test Split for Multi-Class and Binary Classification.

Class ID	Class Type	19 Classes	Train Count (70%)	Test Count (30%)	Binary Classes	Train Count (70%)	Test Count (30%)
0	Normal	300.000	210.504	89.496	300.000	210.504	89.496
1	RDOS	141.261	98.753	42.508	301.498	181.168	120.330
2	Scanning_vulnerability	52.852	36.923	15.929			
3	Generic_scanning	50.277	35.165	15.112			
4	BruteForce	47.241	33.037	14.204			
5	MQTT_cloud_broker_subscription	23.524	16.415	7.109			
6	Discovering_resources	23.148	16.258	6.890			
7	Exfiltration	22.134	15.429	6.705			
8	Insider_malicious	17.447	12.083	5.364			
9	Modbus_register_reading	5.953	4.129	1.824			
10	False_data_injection	5.094	3.565	1.529			
11	C&C	2.863	2.002	861			
12	Dictionary	2.572	1.773	799			
13	TCP Relay	2.119	1.507	612			
14	Fuzzing	1.313	902	411			
15	Reverse_shell	1.016	722	294			
16	Crypto-ransomware	458	322	136			
17	MitM	117	83	34			
18	Fake_notification	28	19	9			
##	Total	601.498	391.672	209.826	601.498	391.672	209.826

4.2. Phase 2: Training and Testing

To address the hybrid nature of IIoT networks, the proposed HADT algorithm incorporates the strengths of both A3C-FFNN and DT with ease. Feature extraction from the input layer involves preprocessing network traffic features, which are then transmitted to a dense layer with 64 neurons and ReLU activation. Multi-class classification (19 classes for class1) is achieved through a softmax activation in the policy output, and the value output provides an estimated state evaluation in the A3C-FFNN algorithm. The DT algorithm provides deterministic decision-making of the probabilities of intrusion classes. Finally, the final predictions are generated by combining outputs from both algorithms and applying additional dense layers. This training process uses an Adam optimizer with a 0.001 learning rate, takes early stops, and requires at least 35 epochs to reduce overfitting. In addition, the algorithm is made more robust and can be generalized by training on the dataset to ensure fair representation across all attack classes.

4.3. Phase 3: Performance Evaluation.

Standard metrics such as accuracy, precision, recall, and F1-score are utilized to measure the performance of the proposed HADT algorithm. The test set is subjected to an evaluation, and the outcomes are compared with baseline algorithms like the A3C-CNN and A3C-FFNN. Accuracy measures the algorithm's overall correctness, while precision denotes the percentage of correctly identified attacks based on all predicted attacks. The ability of the proposed HADT algorithm to identify every attack is assessed by recall, and the F1-score balances both accuracy and recall to provide a comprehensive performance score.

5. Experiments and Results

In this section, the experiment and results of the proposed HADT algorithm for detecting intrusions in the IIoT environment will be explained using multiple experiments on the X-IIoTID dataset. This algorithm was evaluated in comparison to recent A3C models based on CNN and FFNN layers, to demonstrate the effectiveness of detecting a wide range of cyberattack classes with high accuracy.

5.1. Experimental Environment and Hyperparameter Configuration

The HADT algorithm was created using TensorFlow, similar to other DRL experiments currently available. All experiments were conducted using TensorFlow with GPU support. It ran on a 64-bit Windows 10 laptop with 32 GB of RAM, an Intel Core i7-8850 2.70GHz CPU, and an NVIDIA Quadro P3200 GPU. Utilizing the X-IIoTID dataset, the assessments have been carried out. This dataset is considered a benchmark in the context of the IIoT-IDS environment. It also facilitates the comparison of results and supports the evaluation of methods against previous studies.

In this article, the hyperparameters are carefully defined and tuned as listed in Table 3 to enhance the performance of our HADT algorithm for intrusion detection. The number of actions (num_actions = 19) determines the possible decisions the algorithm can take to identify the classes of attacks. Different learning rates were used (0.001 for the A3C-FFNN and 0.0005 for the FFNN) for optimization, which helps balance convergence speed and stability. The neural network layers in A3C workers were trained for 50 epochs with a batch size of 512 to ensure the efficiency of learning. Early stopping was enabled with a patience value of 5 to prevent overfitting. A dense layer with 64 hidden units and a dropout rate of 0.5 was then used for regularization. The A3C-FFNN optimizes the policy loss by using Sparse Categorical Cross-Entropy and the value loss using Mean Squared Error, respectively. Additionally, L2 regularization is applied in the FFNN to prevent overfitting and improve generalization. The hyperparameter configuration of the FFNN in A3C-FFNN and FFNN layers is identified in Table 3.

Table 3.
Hyperparameter Configuration for HADT Algorithm

Parameter	Description	Value
Num of Actions	Number of possible actions in the A3C algorithm	2-Binary Classifications 19-Multiclassification
Learning Rate	Learning rate for optimizer	0.001 (FFNN in A3C), 0.0005 (FFNN)
Epochs	Number of training epochs	50
Batch Size	Number of samples per batch	512
Patience	Early stopping patience	5
Dense Units	Number of neurons in the hidden layer	64
Dropout Rate	Dropout rate for regularization	0.5
Random State	Seeds for reproducibility	42
Test Size	Proportion of the dataset used for testing	0.3
Policy Loss	Loss function for policy output	Sparse Categorical Crossentropy
Value Loss	Loss function for value output	Mean Squared Error
Kernel Regularizer	Regularization for a hybrid layer in the worker	L2

5.2. Evaluation Indicators

This section will make use of the following metrics: 1) A True Positive (TP) is an attack data point that is correctly classified as an attack. Data that is seen as normal yet is incorrectly classified as an attack is known as a false positive (FP). 2) True Negative (TN): Truly classified as normal data even when it is not. 3) A false negative (FN) is attack data that has been incorrectly classified as normal. The effectiveness of this proposed solution will be evaluated using the following metrics:

The precision: Measures the proportion of correctly predicted positive states among all predicted positive states. It is given by Equation (14):

$$\text{Precision} = \frac{TP}{TP + FP} \quad (14)$$

The accuracy determines the proportion of all states that are correctly categorized. It is given by Equation (15):

$$\text{Accuracy} = \frac{TP+TN}{TP + TN + FP + FN} \quad (15)$$

The F1-score is a derived effectiveness statistic that is generated from the harmonic mean of recall and accuracy. It is given by Equation (16):

$$\text{F1-score} = 2 \cdot \frac{\text{P recision} \cdot \text{Recall}}{\text{P recision+Recall}} \quad (16)$$

The recall measures the proportion of correctly predicted positive instances out of all actual positive instances. It is given by Equation (18):

$$\text{Recall} = \frac{TP}{\text{TP} + \text{FN}} \qquad (17)$$

On the other hand, the X-IIoTID dataset was evaluated in two test scenarios: 1) Binary Classification (Normal vs. Intrusion): The dataset was divided into two categories. The HADT algorithm was trained and tested to classify traffic as either Normal or Intrusion. Performance metrics, including accuracy, precision, recall, and F1-score, were computed to assess the detection effectiveness of the HADT algorithm. The experiment also compared the classification performance of A3C-CNN and A3C-FFNN algorithms. 2) Nineteen-Class Classification: The dataset was analyzed with nineteen attack categories, including normal, RDOS, generic scanning, brute force, standard scanning, insider malicious, Modbus register reading, false data injection, C&C, dictionary, TCP relay, fuzzing, reverse shell, crypto-ransomware, MitM, and fake notification. All these nineteen kinds have been identified in this dataset, and classification detection has been carried out. The classification detection produced a nineteen-dimensional confusion matrix (detection result) based on which the accuracy, precision, recall, and F1-score indicators of the HADT for the eighteen assault types were calculated. Using the X-IIoTID dataset with the other A3C networks, which are A3C-CNN and A3C-FFNN, commonly used in intrusion detection techniques, is the second experiment. The experiments carry out testing and training, then compare the test results with the HADT detection results.

5.3. Training Algorithms Results

There were three training and testing procedures in the experiment. Figures 4 to 7 illustrate the relationship between training accuracy and loss during the HADT iterations throughout the training process.

The training and validation accuracy of the FFNN layer in the A3C-FFNN algorithm displayed stability and high performance over the course of 50 epochs, with both values close to 98.5%, as shown in Figure 4.

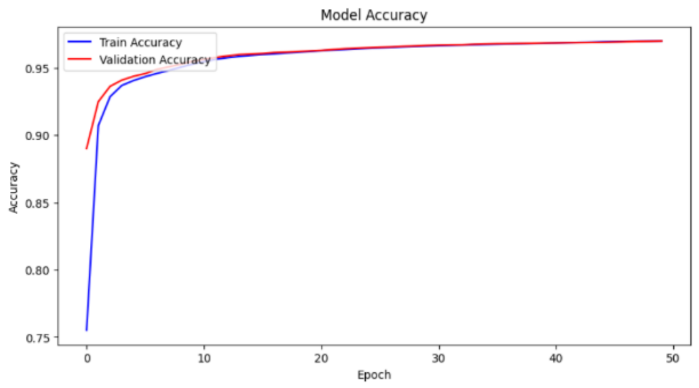

Figure 4.
Training and validation accuracy curves for the A3C-FFNN.

In epoch 35, the training accuracy increases to a constant 0.9842 and then reaches at least an average of 0.9851 in epoch 50, suggesting that this layer is effectively learning from the training data. In the same manner, validation accuracy remains unchanged, with a slight variation between 0.9840 and 0.9847, indicating that the layer can be generalized to this data effectively.

As shown in Figure 5, the figure is not the same when it comes to loss values. At 0.9763, the validation loss is slightly higher than the training loss, which remains steady at 0.8592. Despite the slight difference between train and validation loss, it appears that the layer is well-trained and may require refinement to minimize overfitting. The stable performance of this layer indicates good tuning. These gradually increasing reward values indicate a slight refinement to the layer in making decisions, although further alterations to this layer may improve efficiency.

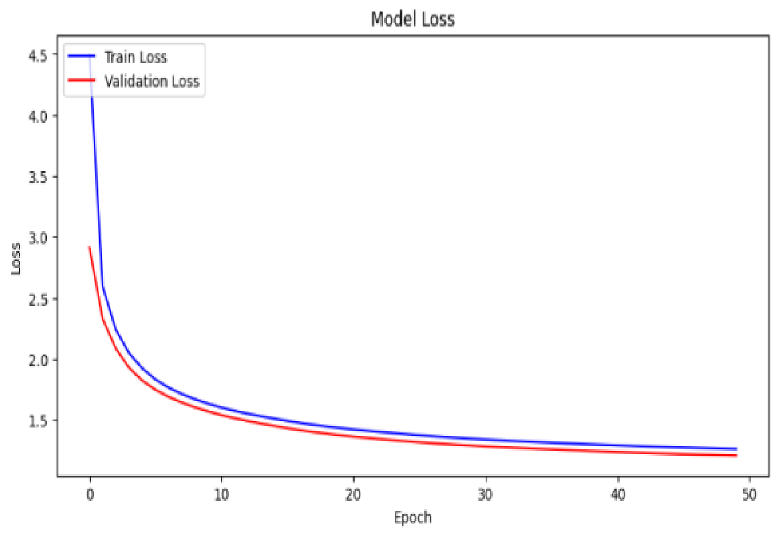

Figure 5.
Training and validation loss curves for the A3C-FFNN.

The FFNN layer, whose input is a combination of the DT output and the A3C-FFNN output, shows significant improvements in both training and validation performance compared to the A3C-FFNN. Across multiple epochs, as shown in Figure 6, the training accuracy of approximately 1.0 indicates that the layer achieved optimal performance on the training data, implying improved overfitting and learning. The validation accuracy also remains very high, at 0.9985 to 0.9986, demonstrating excellent generalization to unseen data, with only minor variations across epochs. Compared to the A3C-FFNN, whose accuracy reached approximately 99.9 percent, this confirms the positive contribution of the hybrid DT to improving the FFNN layer's ability to perform both training and validation tasks with high efficiency.

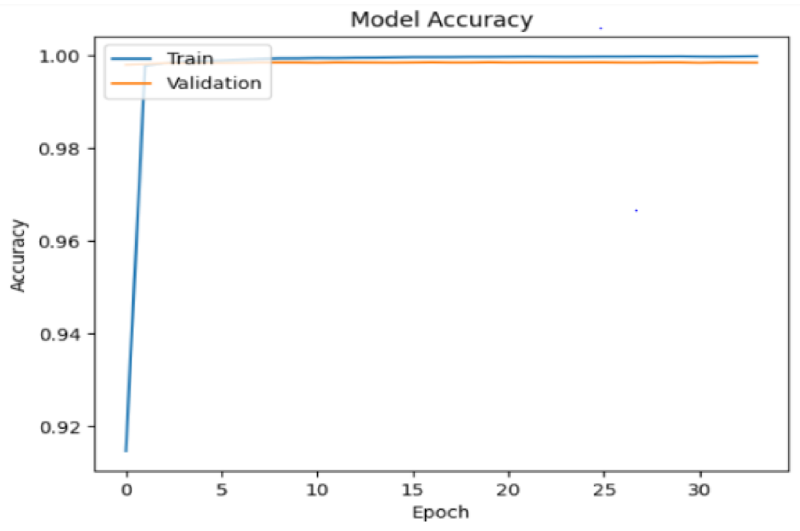

Figure 6.
Training and validation accuracy curves for the FFNN layer.

On the other hand, by epoch 25, as shown in Figure 7, the FFNN layer demonstrates that the training loss steadily decreases from approximately 0.0020 to 0.0009 on the loss curve. This contrasts sharply with the training loss of this layer, which was stable at 0.8592. Meanwhile, the validation loss of FFNN is slightly variable, ranging from 0.0075 to 0.0008, which is significantly lower than the 0.9737 loss achieved by the validated A3C-FFNN. As a result, the FFNN layer improved not only on its training data but also in generalization to the validation set. Overall, by a reasonable margin, the hybrid approach with the DT appears to enhance the layer's accuracy and generalization ability, significantly outperforming the FFNN layer in A3C-FFNN.

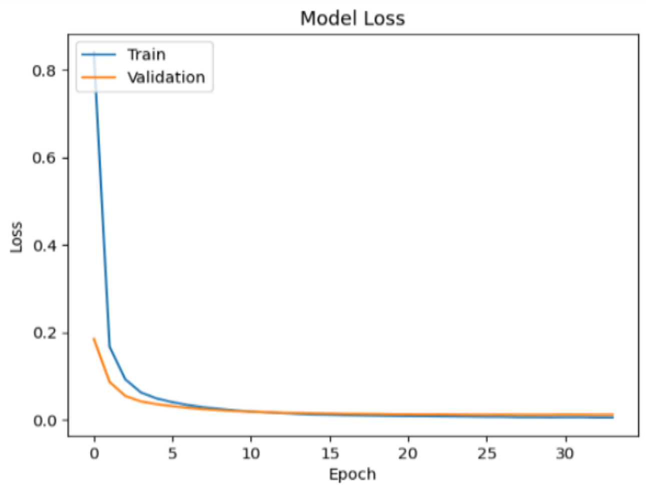

Figure 7.
Training and validation loss curves for the FFNN layer.

5.4. Classification Testing and Results

This section presents the classification performance of the proposed HADT algorithm in both binary and multi-class scenarios comprising 19 classes. The results are evaluated using accuracy (A), precision (P), recall (R), and F1-score (F) to assess the algorithm's effectiveness in detecting various classes of IIoT intrusions.

5.4.1. Binary Classification Results Analysis

The results in binary classification show that the HADT algorithm performs significantly better than both competing algorithms, A3C-CNN and A3C-FFNN, with an accuracy (A), precision (P), recall (R), and F1 score (F), averaging 99.84%, as shown in Table 4.

A3C-CNN outperforms A3C-FFNN on the same metrics. This is due to CNN's superior ability to capture spatial features in IIoT network traffic data, which improves generalization and detection accuracy. Although A3C-FFNN has high performance, it is not as effective as A3C-CNN or the hybrid approach.

Table 4.
The performance comparison of binary classification.

Algorithm	**Binary Classification**			
	A	**P**	**R**	**F1**
A3C-FFNN	98.0123	98.0654	98.0136	98.0353
A3C-CNN	98.5136	98.5345	98.5121	98.5112
HADT	**99.8401**	**99.8408**	**99.8421**	**99.8411**

For applied intrusion detection, the confusion matrix in Figure 8 is crucial for achieving high reliability and minimal misclassification by accurately detecting these results, which are supported by very low false positives (97) and false negatives (238).

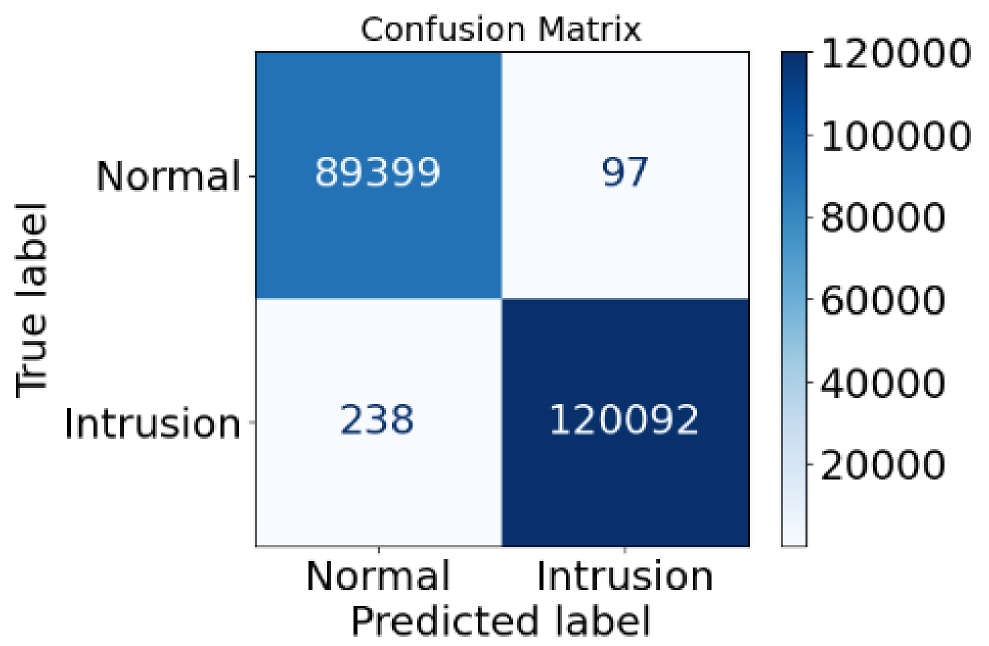

Figure 8.
The confusion matrix of HADT in binary classification.

5.4.2. Multi-Classification Results Analysis

By studying the performance of HADT in the multi-classification case, as shown in Table 5, this algorithm has demonstrated high accuracy across most attack classes, based on the metrics of precision, recall, accuracy, and F1-score, confirming its ability to detect various IIoT attacks.

For common intrusion attacks, all metrics were observed to be above 99.8%, such as Normal (89,496 samples), RDOS (42,508 samples), and Scanning Vulnerability (15,929 samples), which is considered almost ideal detection performance. For intrusion attack classes such as TCP Relay (612 samples), Fuzzing (411 samples), Reverse Shell (294 samples), Crypto-ransomware (136 samples), and MitM (34 samples), it was observed that the precision, recall, and F1-score were approximately equal to 91.17%. Precision and full recall are slightly lower, with 97.1% precision and 100% recall. Overall, HADT is a reliable solution for IIoT security threats, with high precision and generalizability across multiple classes.

Table 5.
The performance of HADT per class.

Class	A	P	R	F	Tot.
Normal	0.998365	0.997289	0.998883	0.998085	89496
RDOS	0.999976	1.000000	0.999882	0.999941	42508
Scanning_vulnerability	0.999938	0.999874	0.999309	0.999592	15929
Generic_scanning	0.999957	0.999735	0.999669	0.999702	15112
BruteForce	1.000000	1.000000	1.000000	1.000000	14204
MQTT_cloud_broker_subscription	0.999971	0.999437	0.999719	0.999578	7109
Discovering_resources	0.999176	0.992521	0.982293	0.987381	6890
Exfiltration	0.999962	0.999851	0.998956	0.999403	6705
Insider_malicious	1.000000	1.000000	1.000000	1.000000	5364
Modbus_register_reading	0.999990	1.000000	0.998904	0.999451	1824
False_data_injection	0.999905	0.992172	0.994768	0.993468	1529
C&C	0.999938	0.990741	0.994193	0.992464	861
Dictionary	1.000000	1.000000	1.000000	1.000000	799
TCP Relay	0.999890	0.975767	0.986928	0.981316	612
Fuzzing	0.999628	0.982609	0.824818	0.896825	411
Reverse_shell	0.999957	0.983051	0.986395	0.984720	294
Crypto-ransomware	0.999995	1.000000	0.992647	0.996310	136
MitM	0.999971	0.911765	0.911765	0.911765	34
Fake_notification	0.999995	0.900000	1.000000	0.947368	9

As shown in Figure 9, the confusion matrix provides a detailed evaluation of the proposed HADT algorithm's classification performance for 19 attack classes in IIoT environments. This algorithm demonstrates strong detection capabilities, with high diagonal values indicating a high rate of correct classifications. Major attack classes such as BruteForce (42,503), MQTT_cloud_broker_subscription (15,918), and Modbus_register_reading (15,107) are classified with near-perfect accuracy. The classification of normal traffic (89,396 instances) also demonstrates the HADT algorithm's effectiveness in distinguishing between normal activity and intrusion attacks. However, minor shortcomings were found in minor attack classes, with some misclassifications. For example, Fake Notification (9 cases) and MitM (31 cases) show minor errors, as well as TCP Relay (612 cases). 17 cases are misclassified, so most likely these errors could be caused by interference with the distribution of features in this dataset due to the possible similarity of some of these features with other types of attacks. These minor errors can be reduced by adding improvements to the datasets, such as data augmentation.

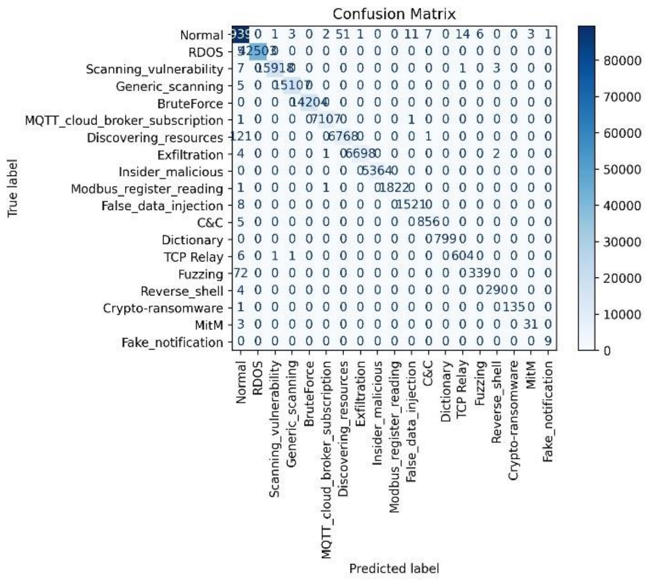

Figure 9.
The confusion matrix of HADT in multi-classification.

The Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC) metrics are examined to evaluate classification performance. By plotting the true positive rate (TPR) and false positive rate (FPR), the ROC curve provides a visual representation of the HADT algorithm's performance across decision thresholds. AUC values as low as 1.0 are indicative of strong classification ability, while those nearing it have higher AUC. The ROC curves for FPR and TPR were generated using sample data from this earlier categorized test from the X-IIoTID dataset. ROC curves are shown as illustrated in Figure 10; the ROC curve fitting degree of the HADT algorithm for the nineteen-classification test is faultless when compared to prior studies' results. AUC values of X-IIoTID, which are between 0.9982 and 1.0000, have been obtained through computation, indicating that the classification algorithm trained on HADT is highly effective in detecting many intrusion types.

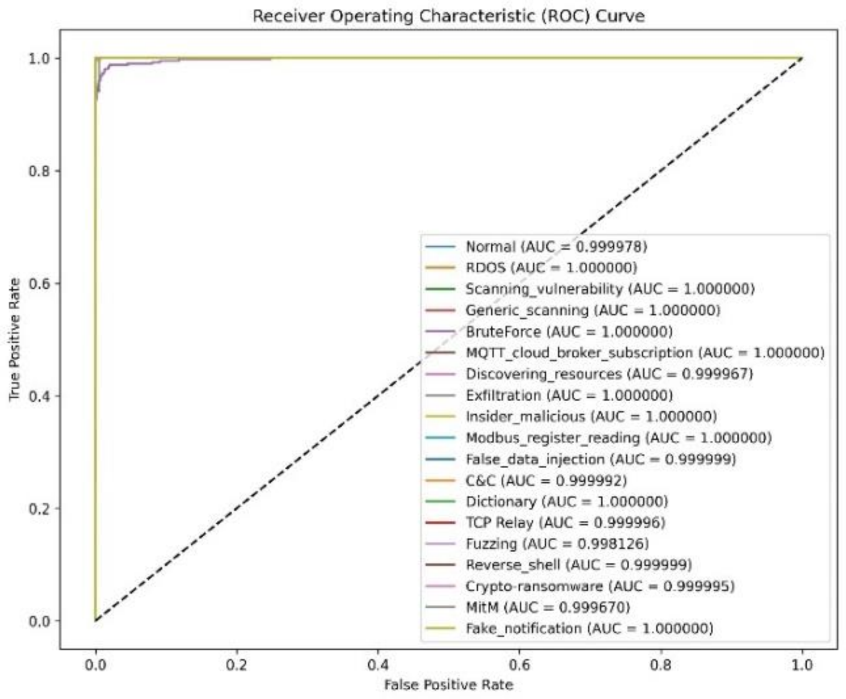

Figure 10.
The HADT nineteen-classification test's ROC curve.

For binary classification, the HADT and current A3C algorithms were used for nineteen classification test training and detection tests. When comparing the test results of Table 6 to the other two algorithms, it is evident that HADT detection has superior accuracy, precision, recall, and F1-score compared to A3C-FFNN and A3C-CNN. These enhanced results are due to the integration of A3C and DT, which combines strengths and reduces each algorithm's weaknesses.

Table 6.
The performance comparison of multi-classification.

Algorithm	**19 Classifications**			
	A	**P**	**R**	**F1**
A3C-FFNN	98.0193	98.0678	98.0124	98.0309
A3C-CNN	98.5186	98.5177	98.5187	98.5156
HADT	99.8305	99.8304	99.8308	99.8291

This table displays the number of test samples from the X-IIoTID dataset, which ensures the equivalence of experimental results. This confirms the superiority of HADT over A3C-FFNN and A3C-CNN in terms of detection accuracy, precision, recall, and an F1-score of approximately 99.83%.

5.4.3. Resource Consumption Analysis

Resource consumption was evaluated based on three factors: training time, inference time, and memory usage. The training time for the HADT algorithm is 726.99 seconds, as shown in Table 7. This makes it more efficient compared to other deeper architectures that are implemented on three layers to achieve the best results, such as A3C-FFNN and A3C-CNN. A3C-FFNN, which consists of three dense layers, was found to require 1013.34 seconds, indicating that increasing the number of dense layers improves the results but significantly impacts the time required to train the algorithm. On the other hand, A3C-CNN, which includes three convolutional layers, had the highest training time of 1552.82 seconds, due to the increased computational complexity of extracting the most important features.

Inference time is particularly important in real-time intrusion detection, enabling rapid decisions to prevent security attacks. HADT's inference time was 28456.32 milliseconds, making it the most efficient algorithm among the evaluated algorithms. A3C-FFNN, due to its additional dense layers, required 41799.95 milliseconds. While the A3C-CNN algorithm takes the longest inference time, at 59382.51 milliseconds, this is primarily due to the computational requirements of the convolutional layers, which process feature maps across multiple dimensions.

Regarding memory usage, HADT is relatively easy to deploy, consuming 3344.69 MB of memory. Despite its denser layers, A3C-FFNN requires 5315.73 MB of storage space, which is significantly more memory than HADT. The A3C-CNN algorithm consumes 7150.31 MB because the convolutional operations and feature maps are memory-intensive. HADT is particularly suitable for resource-constrained edge computing devices.

Table 7.
Resource Consumption Summary.

Layers (Algorithm/Layer)	Training Time (s)	Inference Time (ms)	Memory Utilization (MB)
DT	172.1529	2274.9143	762.4727
FFNN in A3C-FFNN	256.5721	13954.6807	1135.5586
FFNN	297.2711	14226.7299	1446.6602
(DT + FFNN in A3C + FFNN) in HADT	726.9961	28456.3248	3344.6914
(3 Dense Layers) in A3C-FFNN	1023.3483	41799.9536	5315.7334
(3 Conv Layers) in A3C-CNN	1552.8285	59382.5156	7150.3128

5.4.4. Power Consumption

The average power consumption per epoch for the proposed HADT algorithm in an IDS for IIoT at the edge is presented in Table 8. The results highlight the key aspects of processing and memory consumed power. These results demonstrate the feasibility of deploying this algorithm in an edge computing environment.

Table 8.
Power Consumption of HADT.

Processing Power (DT)	Processing Power (A3C-FFNN)	Processing Power (FFNN)	Memory Power (DT)	Memory Power (FFNN in A3C)	Memory Power (FFNN)	Total of Average power/ Epoch
8.595	8.77	8.385	0.48	0.48	0.48	27.19 W

The processing power of HADT is divided into three main components. On average, the DT component consumes 8.595 watts. The A3C-FFNN requires slightly more processing power at 8.77 watts. The average power consumption of the hybrid processing layer (the FFNN layer), which receives outputs from both DT and A3C, is 8.385 watts, indicating a better balance between decision-making and learning capability. Furthermore, memory consumes a constant 0.48 watts across all components. As shown in Figure 11, the average total power consumption of HADT per epoch is 27.19 watts, which is reasonable given the complexity of real-time intrusion detection at the edge. While A3C-based CNN and FFNN consume more power, A3C-CNN, in particular, consumes the most (52.57 watts) due to its deep convolutional layers and the high computational cost of feature extraction and training.

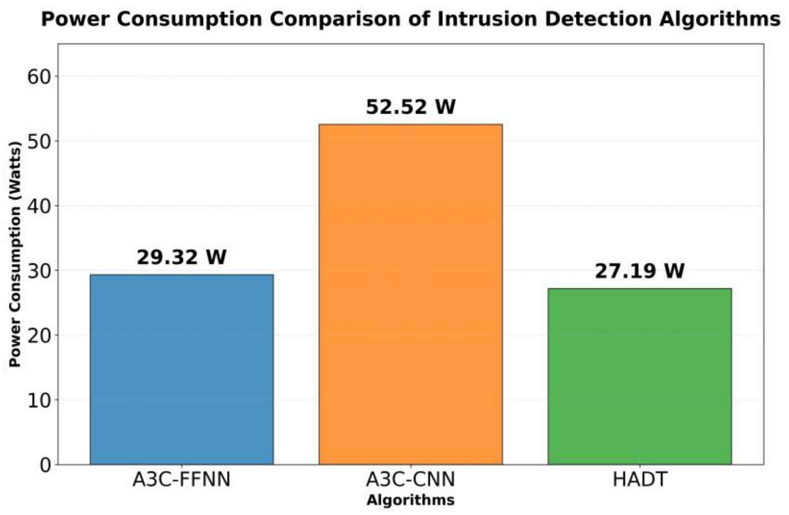

Figure 11.
A comparison of the power consumption of the algorithms.

It is noted that the A3C-FFNN consumes 29.32 watts of power, which is moderate compared to the A3C-CNN (52.52 watts). This is due to the fact that the computational costs of the A3C-FFNN are lower than those of the CNN layers in A3C-CNN. However, the A3C-FFNN cannot extract spatial dependencies in the IIoT network, which can affect the sequence of intrusion detection performance.

On the other hand, the HADT consumes 27.19 watts. Therefore, this algorithm can provide high accuracy in intrusion detection while maintaining acceptable power consumption, making it suitable for practical implementation in IDS.

6. Conclusion

This article proposes the HADT algorithm, which integrates two hybrid algorithms: A3C-FFNN and DT to improve intrusion detection in the IIoT environment. This HADT algorithm, which uses fewer neural network layers to reduce computational overhead and improve processing time and energy efficiency, addresses the challenges that exist in the IIoT environment, including different attack patterns of a stochastic or deterministic nature. This is due to the presence of features with continuous and discrete values in the X-IIoTID dataset. Therefore, the proposed algorithm leverages the integration advantages of DT by dealing with data from a deterministic environment. In addition, the simple A3C is more suitable to be applied in a stochastic environment using the FFNN layer in its worker to reduce the consumption power due to its simplicity in calculating the processing during training and testing.

The HADT algorithm outperformed the current A3C algorithms, including A3C-FFNN and A3C-CNN, achieving better results with weighted accuracy, precision, recall, and F1-scores exceeding 99.8% across both binary and multi-class classifications.

In addition, the proposed HADT displays remarkable proficiency in identifying low sample sizes of attack classes, effectively alleviating a significant defect in current systems. The results show that the proposed HADT algorithm is more effective in identifying different classes of attacks, making it a strong algorithm for IIoT security.

Despite the strong performance of this algorithm, it faces some limitations. Therefore, future studies should focus on improving energy consumption and adaptive learning to increase the efficiency in detecting and mitigating zero-day attacks in real-time, especially with the continuous expansion of IIoT networks with an increasing number of data pattern classes. Overall, this work makes a significant contribution to the security of IIoT, paving the way for IDSs in edge computing to be more accurate and intelligent in this complex environment.

Transparency:

The authors confirm that the manuscript is an honest, accurate, and transparent account of the study; that no vital features of the study have been omitted; and that any discrepancies from the study as planned have been explained. This study followed all ethical practices during writing.

Acknowledgment:

The author would like to express sincere gratitude to all those who provided support, guidance, and encouragement throughout the completion of this research. Special appreciation is extended to Al-Ghad College of Technical Health Sciences for its continuous support.

Copyright:

References

[1] A. Guezzaz, M. Azrour, S. Benkirane, M. Mohy-Eddine, H. Attou, and M. Douiba, "A lightweight hybrid intrusion detection framework using machine learning for edge-based IIoT security," *International Arab Journal of Information Technology*, vol. 19, no. 5, pp. 822–830, 2022. https://doi.org/10.34028/iajit/19/5/14

[2] M. A. Q. Al-Riyashi, A. T. Zahary, and F. Saeed, "A review for Fog-Cloud Security: aspects, attacks, solutions, and trends," *Sana'a University Journal of Applied Sciences and Technology*, vol. 2, no. 5, pp. 482-491, 2024. https://doi.org/10.59628/jast.v2i5.1128

[3] S. Abdelkader *et al.*, "Securing modern power systems: Implementing comprehensive strategies to enhance resilience and reliability against cyber-attacks," *Results in engineering*, vol. 23, p. 102647, 2024. https://doi.org/10.1016/j.rineng.2024.102647

[4] A. Y. Abohatem, F. M. M. Ba-Alwi, and A. A. Al-Khulaidi, "Suggestion cybersecurity framework (CSF) for reducing cyber-attacks on information systems," *Sana'a University Journal of Applied Sciences and Technology*, vol. 1, no. 3, pp. 234–252, 2023. https://doi.org/10.59628/jast.v1i3.248

[5] F. Thabit, S. Alhomdy, and S. Jagtap, "A new data security algorithm for the cloud computing based on genetics techniques and logical-mathematical functions," *International Journal of Intelligent Networks*, vol. 2, pp. 18-33, 2021. https://doi.org/10.1016/j.ijin.2021.03.001

[6] H. Kheddar, D. W. Dawoud, A. I. Awad, Y. Himeur, and M. K. Khan, "Reinforcement-learning-based intrusion detection in communication networks: A review," *IEEE Communications Surveys & Tutorials*, vol. 27, no. 4, pp. 2420 -2469, 2024. https://doi.org/10.1109/COMST.2024.3484491

[7] F. Zhang, H. Huang, H. Lv, M. Jia, and J. Liu, "Soft actor-critic based anti-attack XSS detection," in *2024 IEEE 24th International Conference on Software Quality, Reliability, and Security Companion (QRS-C) (pp. 591-600). IEEE, 2024.*

[8] N. Niknami and J. Wu, "DeepIDPS: An adaptive DRL-based intrusion detection and prevention system for SDN," in *ICC 2024-IEEE International Conference on Communications (pp. 2040-2046). IEEE, 2024.*

[9] J. Watts, F. Van Wyk, S. Rezaei, Y. Wang, N. Masoud, and A. Khojandi, "A dynamic deep reinforcement learning-Bayesian framework for anomaly detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 22884-22894, 2022. https://doi.org/10.1109/TITS.2022.3200906

[10] S. Mishra, S. Chakraborty, K. S. Sahoo, and M. Bilal, "Cogni-Sec: A secure cognitive enabled distributed reinforcement learning model for medical cyber–physical system," *Internet of Things*, vol. 24, p. 100978, 2023. https://doi.org/10.1016/j.iot.2023.100978

[11] D. Lim and I. Joe, "A DRL-based task offloading scheme for server decision-making in multi-access edge computing," *Electronics*, vol. 12, no. 18, p. 3882, 2023. https://doi.org/10.3390/electronics12183882

[12] A. S. Rajawat, S. Goyal, C. Chauhan, P. Bedi, M. Prasad, and T. Jan, "Cognitive adaptive systems for industrial internet of things using reinforcement algorithm," *Electronics*, vol. 12, no. 1, p. 217, 2023. https://doi.org/10.3390/electronics12010217

[13] M. H. Ali *et al.*, "Threat analysis and distributed denial of service (DDoS) attack recognition in the internet of things (IoT)," *Electronics*, vol. 11, no. 3, p. 494, 2022. https://doi.org/10.3390/electronics11030494

[14] A. M. Alnajim, S. Habib, M. Islam, S. M. Thwin, and F. Alotaibi, "A comprehensive survey of cybersecurity threats, attacks, and effective countermeasures in industrial internet of things," *Technologies*, vol. 11, no. 6, p. 161, 2023. https://doi.org/10.3390/technologies11060161

[15] M. Nuaimi, L. C. Fourati, and B. B. Hamed, "Intelligent approaches toward intrusion detection systems for Industrial Internet of Things: A systematic comprehensive review," *Journal of Network and Computer Applications*, vol. 215, p. 103637, 2023. https://doi.org/10.1016/j.jnca.2023.103637

[16] 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) (pp. 106-113). IEEE*, 2021.

[17] K. Zhou, W. Wang, T. Hu, and K. Deng, "Application of improved asynchronous advantage actor critic reinforcement learning model on anomaly detection," *Entropy*, vol. 23, no. 3, p. 274, 2021. https://doi.org/10.3390/e23030274

[18] S. Shen, C. Cai, Y. Shen, X. Wu, W. Ke, and S. Yu, "Joint mean-field game and multiagent asynchronous advantage actor-critic for edge intelligence-based IoT malware propagation defense," *IEEE Transactions on Dependable and Secure Computing*, vol. 22, no. 4, pp. 3824 - 3838, 2025. https://doi.org/10.1109/TDSC.2025.3542104

[19] S. Dong, Y. Xia, and T. Wang, "Network abnormal traffic detection framework based on deep reinforcement learning," *IEEE Wireless Communications*, vol. 31, no. 3, pp. 185-193, 2024. https://doi.org/10.1109/MWC.011.2200320

[20] S. Anbazhagan and R. Mugelan, "Next-gen resource optimization in NB-IoT networks: Harnessing soft actor–critic reinforcement learning," *Computer Networks*, vol. 252, p. 110670, 2024. https://doi.org/10.1016/j.comnet.2024.110670

[21] A. M. Khako and S. A. Alhomdy, "Enhanced A3C with modified CNN for intrusion detection in IIoT environments," *Journal of Cybersecurity and Network Protection*, vol. 5, no. 2, pp. 112-125, 2023.

[22] Z. Hu *et al.*, "A hybrid data-driven approach integrating temporal fusion transformer and soft actor-critic algorithm for optimal scheduling of building integrated energy systems," *Journal of Modern Power Systems and Clean Energy*, vol. 13, no. 3, pp. 878 - 891, 2025. https://doi.org/10.35833/MPCE.2024.000909

[23] H. Kumar, A. Koppel, and A. Ribeiro, "On the sample complexity of actor-critic method for reinforcement learning with function approximation," *Machine Learning*, vol. 112, pp. 2433-2467, 2023. https://doi.org/10.1007/s10994-023-06303-2

[24] J. Zheng, M. N. Kurt, and X. Wang, "Stochastic integrated actor–critic for deep reinforcement learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 5, pp. 6654-6666, 2022. https://doi.org/10.1109/TNNLS.2022.3212273