Scientific Research Publishing

International Journal of Intelligence Science, 2025, 15(4), 263-278
https://www.scirp.org/journal/ijis
ISSN Online: 2163-0356
ISSN Print: 2163-0283

Intelligent Approach to Intrusion Detection Based on Multi-Agent Reinforcement Learning (MARL) in Smart IoT Environments

Nguendap Severin[1], Fotsing Eric[2], Hamdane Allamine Moussa[1], Kakeu Tuekam Severin Vianey[1], Nzeukou Takougang Armand[3], Lienou Jean Pierre[1]

[1]Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon
[2]Department of Computer Engineering, IUT-FV, University of Dschang, Bandjoun, Cameroon
[3]Department of Physic, University of Dschang, Dschang, Cameroon
Email: snguendap@gmail.com, efotsing@gmail.com, severinkakeu@gmail.com, armand.nzeukou@univ-dschang.org, jp.lienou@univ-dschang.org

How to cite this paper: Severin, N., Eric, F., Moussa, H.A., Severin Vianey, K.T., Armand, N.T. and Pierre, L.J. (2025) Intelligent Approach to Intrusion Detection Based on Multi-Agent Reinforcement Learning (MARL) in Smart IoT Environments. *International Journal of Intelligence Science*, **15**, 263-278.
https://doi.org/10.4236/ijis.2025.154011

Received: August 18, 2025
Accepted: October 8, 2025
Published: October 11, 2025

Abstract

This work presents a new method to improve the security of Internet of Things (IoT) networks using a model combining an autoencoder LSTM (AE-LSTM) and a centralized multi-agent reinforcement learning (RL) system. IoT, essential in fields such as healthcare, smart cities, or agriculture, is facing increasing threats (DDoS attacks, intrusions, botnets like Mirai or Gafgyt) due to its often poorly protected connected objects. Traditional intrusion detection methods fail in the face of complex attacks, which has led to the exploration of artificial intelligence techniques. The AE-LSTM model extracts temporal features from network data, while the multi-agent reinforcement learning (RL) system, based on the PPO algorithm, enables collaborative and adaptive anomaly detection using common latent vectors. The IoT-23 dataset, realistic and captured in an IoT environment, was carefully prepared: data cleaning (removing missing and duplicate values), reducing high correlations, normalization, and class balancing. Experimental results show that the model achieves a precision of 99.30%, a recall of 99.35%, and an F1-score of 99.40%, with a reward curve associated with the PPO algorithm that shows constant progression during training, demonstrating stable convergence of the learning process. This approach outperforms other existing methods in terms of robustness and generalization and provides a solid foundation for real-time applications. The reward curve associated with the PPO algorithm shows constant progression during training, demonstrating stable convergence of the learning process. This approach outperforms other existing methods in terms of robustness and generalization and provides a solid foundation for real-time applications.

Keywords

IoT Environments, Intrusion Detection, AE-LSTM, Reinforcement Learning, Multi-Agent, PPO, IoT-23

1. Introduction

Smart IoT environments represent an opportunity in digitalization today, interconnecting multiple devices that collect, process, and exchange data in real time. These devices, combined with a multitude of communication protocols and architectures, and limited resources, create complex and dynamic environments, particularly vulnerable to cyber threats from all sides. In this context, the intrusion management system (IMS) is becoming a crucial issue for ensuring the cybersecurity, availability, and resilience of smart IoT environments.

Modeling the temporal and nonlinear behaviors of IoT systems remains a major challenge, and the solution based on artificial intelligence approaches, particularly deep learning (DL), represents an opportunity and a powerful tool capable of learning, adapting, and detecting abnormal behaviors in network traffic. Among these approaches, autoencoders (AE) acquire the ability to compress input data into a low-dimensional latent space, and then reconstruct the initial data from this latent representation, and minimize the dissimilarity between the input data and their reconstructed counterpart. However, they encounter difficulties when processing long sequences, specifically those that include more than several events. Thanks to the LSTM model, the model is able to maintain a state over long periods of time if necessary, and allows both extracting the internal characteristics of the sequences and capturing their temporal dynamics. The LSTM encoder learns to capture sequential dynamics and long-term dependencies, while the decoder reconstructs the input sequence, thus identifying significant deviations that indicate anomalies. Furthermore, the model requires coordinated and adaptive decision-making mechanisms. This is where Multi-Agent Reinforcement Learning (MARL) becomes particularly relevant. In a MARL system, multiple agents—deployed at different nodes or sub-networks—interact with their local environment, share information, and learn optimal decision-making policies based on defined rewards. Integrating an AE-LSTM module into MARL agents not only equips each agent with advanced sequential perception for anomaly detection, but also globally optimizes resource management, coordination, and threat response.

Thus, the combination of AE-LSTM + multi-agent RL paves the way for intelligent IoT monitoring systems capable of distributed learning, online adaptation, and collaborative decision-making. This synergy makes it possible to reconcile detailed local analysis with a shared global vision, offering a compromise between accuracy, scalability, and robustness, key elements for addressing the security and performance challenges of the next-generation Internet of Things.

In our work, we propose a hybrid AE-LSTM model combined with a centralized

multi-agent reinforcement learning (RL) system. This innovative architecture relies on centralized coordination where multiple agents (agent 0 to agent 4) share a common observation derived from the latent vectors generated by the AE-LSTM. Each agent makes an individual decision (binary action: 0 or 1), but the reward and observation logic are managed in a unified manner via a centralized environment, allowing prediction fusion to produce a final majority label. By leveraging a single PPO model trained with Stable-Baselines3, this approach ensures adaptive and collaborative anomaly detection without requiring decentralized communication between agents. To evaluate this solution, we use the IoT-23 dataset, a realistic dataset captured in an IoT environment, faithfully reflecting current IoT traffic conditions with high heterogeneity, varied anomalies, and real-world infection scenarios.

This work is part of this dynamic and aims to contribute to the development of intelligent, data-driven approaches for intelligent intrusion management in IoT environments. The current work was designed to combine three promising techniques (autoencoders, LSTM, and multi-agent reinforcement learning) in a novel and optimized way to predict attacks and ensure the security of IoT environments. The main objective is to develop an AE-LSTM model combined with multi-agent reinforcement learning to efficiently detect anomalies in IoT networks from the IoT-23 dataset, exploiting the temporal characteristics and dynamic behaviors of network traffic.

Two specific objectives guide this study:

1) Prepare and optimize the IoT-23 dataset to extract robust features, eliminating missing data and high correlations, to ensure a reliable basis for training the multi-agent AE-LSTM + RL model.

2) Design a multi-agent RL environment leveraging AE-LSTM latent representations to enable adaptive and collaborative anomaly detection, such as DDoS attacks and malicious intrusions, in real-world IoT scenarios.

This article is organized as follows: Section II provides a literature review. Section III discusses the proposed methodology. Section IV experiments with the solution on the dataset. Section V provides results and discussion, and Section VI provides conclusions and outlook.

2. State of the Art

This section provides an overview of intrusion detection approaches in the IoT, focusing on the main concerns related to IoT environments, the limitations of traditional systems, and the contributions of intelligent systems. It also proposes the advantages of intelligent data-driven approaches. Smart IoT environments include a wide and diverse range of resource-constrained devices and varying communication standards. This heterogeneity complicates traditional security mechanisms. IoT architectures often follow layered models, and each presents distinct vulnerabilities. Many IoT communication protocols, such as MQTT, CoAP, AMQP, and DDS, were designed to be lightweight with robust security in mind. The IMS

must be capable of detecting and responding in real time. The use of reinforcement learning (RL), and particularly deep Q-learning, in network intrusion detection systems is the main contribution of the framework proposed in Kim2019. The authors use two deep autoencoders in their RL framework. One is used to train the Q-learning model, and the other is used to update it. The framework periodically applies mini-batch or Q-learning updates to make the model more adaptable to the continuous evolution of cyberattacks.

Mahdi *et al.* [1] propose a framework to adapt DL-based models to the changing behavior of attack/benign traffic by considering a more practical scenario (*i.e.*, online adaptive IDS). They evaluate the proposed framework using different deep models (including CNN- and LSTM-based) on the CIC-IDS2017 and CSE-CIC-IDS2018 datasets. The results indicate that while both architectures perform well, CNN models win in terms of an average detection rate above 95% while only needing 128 new flows for the update phase, and LSTM models are more suitable for early anomaly detection with only a few packets. Amani Bacha *et al.* introduce a decentralized approach based on the concept of Vertical Feederated Reinforcement Learning (VFRL), as proposed by Qi *et al.* (2021). The developed models, COCA-MADQN and MADQN-GTN, demonstrate notable robustness in the face of non-stationarity and scalability challenges in distributed environments. However, the computational cost associated with these architectures remains a determining factor to consider when choosing a solution adapted to practical constraints.

Mehdi Kheira *et al.* [2] proposed an intrusion detection system based on a decentralized architecture through a multi-agent system. They use machine learning methods: decision trees, SVM, and neural networks to facilitate the monitoring of the network's security status in order to be able to respond effectively in the event of detected intrusions. Communication between agents allows real-time verification of the status of the system's various agents.

Mahmoud *et al.* [3] propose an AE-LSTM model designed to detect intrusions in IoT networks, leveraging the NSL-KDD dataset. The model is structured around a six-layer deep autoencoder, combined with LSTM layers to capture the temporal structure of network flows. To improve training reliability, the authors introduced a preprocessing method to eliminate outliers, which are often the cause of data imbalance. The model was evaluated in both binary (normal vs. malicious traffic) and multiclass (DoS, Probe, R2L, U2R, Normal) classification. The results show an accuracy of 98.88% and an equivalent F1-score, confirming the robustness of this architecture for flow analysis in controlled environments.

Xu *et al.* [4] used an autoencoder (AE)-based model and a novel outlier removal method to avoid bias due to data imbalance for each input sample. They achieved an accuracy of 90.61% and an F-score of 92.26% on the NSL-KDD test dataset. Su *et al.* [5] proposed the BAT method. It consists of BLSTM (Bidirectional-LSTM) and attention methodology, achieving an accuracy of 84.25% on the NSL-KDD test dataset. Wang and Li [6] developed a hybrid neural network framework (DosTC) that combines efficient and scalable transformers with a CNN (Convolutional

Neural Network) to detect DDoS (Distributed Denial of Service) attacks on SDN, which was evaluated on the CICDDoS219 dataset.

Elsayer *et al.* [7] proposed DDoSNET to detect distributed denial of service (DDoS) attacks in Software Defined Networks (SDN). They leveraged deep learning (DL) to develop their method, which is based on a recurrent neural network (RNN) with an autoencoder. They evaluate their model on the CICDDoS2019 dataset.

After establishing this framework, we move on to the methodological description of our own approach in the next section.

3. Methodology

In this section, we discuss the methodology used to conduct our project. We begin by describing the dataset used, highlighting its specific characteristics and the data preprocessing applied. We then present the model architecture and the algorithms we selected to implement our solution.

3.1. Datasets Used

The IoT-23 dataset [8] is a new dataset of network traffic from Internet of Things (IoT) devices. It contains 20 captures of malware running on IoT devices and three captures of benign IoT device traffic. It was first published in January 2020, with captures spanning from 2018 to 2019. This IoT network traffic was captured in the Stratosphere Lab, AIC Group, FEL, CTU University, Czech Republic. Its goal is to provide a large dataset of real-world, labeled IoT malware infections and benign IoT traffic for researchers to develop machine learning algorithms. This dataset and its research are funded by Avast Software, Prague.

3.2. Data Preprocessing

Data preprocessing is a crucial step in our architecture, combining an LSTM autoencoder (AE-LSTM) and a multi-agent reinforcement learning system (Multi-Agent PPO), applied to intrusion detection on the IoT-23 dataset.

The IoT-23 dataset consists of conn.log.labeled files, each corresponding to a specific network scenario (e.g., Mi-rai, infiltration, normal activity, etc.). To ensure optimal quality of the inputs processed by the autoencoder and RL agents, several preparation steps are necessary.

First, rigorous cleaning is performed. This consists of deleting rows without a detailed label (detailed-label), eliminating missing values (NaN), infinite values (Inf), and duplicates. A type conversion is then applied to transform all categorical columns into usable numeric formats. In addition, a correlation matrix analysis is performed, and all highly correlated columns (correlation > 0.95) are removed to avoid redundancy and reduce dimensionality.

Next, manual label mapping is performed to associate each type of network activity with a unique numeric identifier. This allows the autoencoder to distinguish between normal and abnormal behaviors and facilitates the interaction of RL agents with their environment based on rewards per type of detected behavior.

Normalization is then applied to all numeric variables using the StandardScaler method to center and reduce the data, ensuring stable convergence of the AE-LSTM model.

Finally, to ensure balanced exploration of the environment by the PPO agents, minority classes are resampled. Each class is balanced to contain a fixed number of examples (e.g., 1500) to ensure each PPO agent has a fair representation of the various network states to be analyzed. The final dataset, thus cleaned, converted, normalized, and balanced, is then transformed into time sequences to be injected into the AE-LSTM module, which is used to encode the dynamics of network flows. These representations are then used as input states for the PPO agents responsible for making optimal decisions for anomaly or attack detection.

1) **Preprocessing Process Algorithm (Figure 1):**

```
\begin{algorithm}[H]
\caption{Preprocessing\_AE\_LSTM\_RL\_MultiAgent}
\KwIn{
    $D_{total}$: Ensemble complet de données brutes (e.g., \texttt{cons.log.labeled}) \\
    $N$: Nombre de fenêtres par classe après rééquilibrage (e.g., 1500) \\
    $corr_{threshold}$: Seuil de corrélation pour l'élimination de colonnes redondantes (e.g., 0.95)
}
\KwOut{
    $D_{seq}$: Données séquentielles normalisées pour l'entraînement de l'AE-LSTM et des agents RL
}

$D_{total} \leftarrow \emptyset$ \\
\ForEach{fichier $f$}{
    Charger $f$ dans un DataFrame $df$ \\
    Supprimer les lignes de connexions vides, erronées ou non initialisées de $df$ \\
    Supprimer les colonnes non pertinentes (e.g., identifiants techniques) de $df$ \\
    Ajouter $df$ à $D_{total}$
}

Calculer la matrice de corrélation $Corr$ sur $D_{total}$ \\
Supprimer les colonnes fortement corrélées ($> corr_{threshold}$) \\
Appliquer la normalisation des labels (e.g., \texttt{normal} $\mapsto$ 0, \texttt{mirai} $\mapsto$ 1, etc.) \\
Appliquer \texttt{StandardScaler()} à toutes les colonnes numériques \\

\ForEach{classe $C$ dans $D_{total}$}{
    \eIf{$|C| < N$}{
        Sur-échantillonner $C$ avec remise jusqu'à obtenir $|C| = N$
    }{
        Sélectionner aléatoirement $N$ échantillons de $C$
    }
}

Fusionner toutes les classes rééquilibrées dans $D_{balanced}$ \\
Mélanger $D_{balanced}$ aléatoirement \\
Transformer $D_{balanced}$ en séquences temporelles de longueur fixe $T$ pour obtenir $D_{seq}$ \\
\Return{$D_{seq}$}
\end{algorithm}
```

Figure 1. Preprocessing and data cleaning process algorithm.

3.3. AE-LSTM + Multi-Agent RL

Figure 2 illustrates the different steps of the proposed process for detecting intrusions in IoT networks. The system is based on a combination of an LSTM autoencoder for feature extraction and a set of PPO (Proximal Policy Optimization)

agents operating in a multi-agent reinforcement learning environment. This innovative architecture is designed to learn optimal behaviors in response to different types of IoT traffic, based on the compact representations provided by the AE-LSTM model.

The processing of the IoT-23 dataset uses captures based on complete traffic scenarios (Pcap, Flows) rather than independent samples. To prevent data leakage during anomaly detection or classification, the following steps should be taken:

— Rigorously divide the data into two groups (80% training, 20% testing) while ensuring that the test set remains clean and independent;
— Transform flows into usable time series generated within a scenario for AE-LSTM for a window of 100 packets or 30 seconds of flow, and ensure that all windows in the scenario belong to the same split (Train, Test);
— Balance the classes without breaking the train/test separation to prevent Mirai from flooding the training and normalise the proportions after performing the train/test separation.

The splits were performed at the level of entire scenarios to avoid any data leakage when processing the IoT-23 dataset. The flows and time windows extracted from a given scenario are assigned exclusively to the training/validation set, or to the test set, but never to both simultaneously. After this strict separation, the sessions are windowed into fixed sequences after separation, and then resampling is applied to ensure a balance between classes, particularly between normal and malicious traffic. This approach ensures that the test set remains clean, allowing for reliable measurement of the model's ability to generalise to new scenarios and IoT devices never seen during training.

3.4. Model Selection and Justification

The AE-LSTM + RL Multi-agent offers both the power of unsupervised reconstruction by detecting anomalies through reconstruction errors, temporal modelling with LSTM, which is a double advantage over zero-day attacks and specific attacks, and distributed, adaptive, and cooperative decision-making thanks to multi-agent RL.

The autoencoder (AE) learns a compact latent representation of normal traffic by reducing dimensions and filtering noise. This latent compression allows anomalies to be detected effectively by comparing the input with its reconstruction. LSTMs are capable of memorising long-term dependencies and managing the gradient effect, making them suitable for detecting anomalies in IoT flows.

Once the features have been extracted by the AE-LSTM, decision-making is entrusted to a multi-agent reinforcement system. Each agent corresponds to an IoT node or IoT traffic source and shares a common PPO policy, but each has its own local observations, learning to cooperate via a centralized reward.

This model is suitable for smart IoT environments because it allows actions (block, alert, ignore) to be dynamically adapted, optimises the overall network strategy (minimising false positives, maximising attack coverage), and coordinates multiple

IoT entities for distributed cybersecurity.

It is suitable for massive and distributed IoT environments, detects unprecedented anomalies (zero-day attacks) while the RPs learn to adapt their responses.

3.5. Features of the AE-LSTM + RL Multi-Agent Model

The encoder is a 3-layer LSTM with a hidden layer size of 64 units. It transforms each input sequence into a compact latent vector representing the essential characteristics of network traffic. Only the last hidden layer (h_n [−1]) is used as the final representation. The latent vector x_enc = h_[−1] corresponds to the last hidden layer of the LSTM encoder. It is replicated at each time step to form the decoder input. This vector encodes the global summary of the sequence. Decoder (LSTM): It attempts to reconstruct the original sequence from the summary obtained. If the reconstruction is poor (using a measure called MSE), this may indicate an anomaly. The decoder is also a 3-layer LSTM, but inverted: it takes the latent vector (replicated) as input and attempts to reconstruct the original sequence. It measures the model's ability to memorise and reconstruct traffic behavior. The reconstruction loss is calculated using the MSE-Loss (Mean Squared Error) function. It measures the difference between the original sequence and the sequence reconstructed by the decoder. A faithful reconstruction means that the sequence is 'normal', while a high difference may indicate an anomaly or an attack.

RL Multi-agent is based on the CTDE (Centralised Training, Decentralised Execution) paradigm: All agents share a common PPO policy and are trained with a global reward to stabilise optimisation. Each agent observes a different portion of the traffic (local flows, metrics specific to its context) and applies a policy to its own local observation, enabling distributed detection in the IoT environment. Even if the policy is identical, local observations differ, generating heterogeneous behaviours depending on the case.

AE-LSTM + Multi-Agent RL Ablation: The model is trained using the Adam optimiser with the following parameters:

— Dropout: 03.
— Weight decay 1e−4. Limits overfitting and improves generalisation.
— Windows: size (T = 60), stride (non-overlapping).
— Total loss function: MSE + CrossEntropy, implicitly weighted equally.
— The threshold is chosen by maximising the metrics (F1-score, precision, recall, AUROC) on a validation set containing normal and attack examples.
— Bottleneck: latent 32. The model learns to capture only what is essential.

Comparisons: In order to evaluate the relevance of using PPO for intrusion detection in IoT environments, we compare our approach with several representative alternatives: (a) autoencoder (AE) coupled with a simple classifier, (b) supervised sequential models (LSTM, GRU, Transformer), (c) PPO agent alone or combined with an encoder, and (d) single-agent or multi-agent variants.

(a) Autoencoder (AE) + Simple Classifier: The AE extracts latent representations (bottlenecks) from IoT environments, and the classifier is responsible for using them to predict attacks. This configuration allows for effective dimension-

ality reduction and limits the risk of overfitting, while remaining relatively explainable. However, it relies heavily on the quality of the representations learned by the AE and may lack the ability to capture complex temporal dependencies in network flows.

(b) Supervised Sequential Models: LSTM, GRU, or Transformer networks are trained directly to classify sequences as normal or malicious. They are particularly well-suited for modelling long-term temporal dependencies and perform excellently on rich datasets such as IoT-23. However, they require a large amount of labelled data and are sensitive to zero-day attacks, which are not represented in the training data. Furthermore, their high capacity exposes them to the risk of overfitting in low-data-volume contexts and requires careful encoding and often full supervision.

(c) Reinforcement Learning with PPO: The use of PPO reformulates detection as a sequential decision-making problem, where the agent learns a reward-guided classification policy. This flexibility makes it possible to minimise the false positive rate or give greater weight to false negatives. Without an encoder, the agent learns directly from the normalised features, which simplifies the pipeline but can be inefficient in large state spaces. With an AE, latent representations enhance robustness but at the cost of increased complexity.

(d) Single Agent vs. Multi-Agent: In IoT environments, multi-agent architectures are often preferred for their ability to react locally while sharing global signals. Multi-agent systems can provide increased robustness through majority voting for agents that share the same policy. A single PPO agent is a simple and effective configuration, while a multi-agent approach aims to increase robustness through policy diversity or a majority voting mechanism. Centralised training with decentralised execution (CTDE) is an interesting variant, where multiple agents share a global reward during training but then make decisions independently. However, the multi-agent configuration involves additional complexity, and if the agents share the same policy, the gain over a single agent may remain limited.

Experimentation scripts: The entire process is organized into modular Python scripts:

1) preprocess.data.py: processing, transformation, and export of sequences.

2) train_ae_lstm.py: training of the AE-LSTM (latent, dropout, epochs, batch-size, learning_rate).

3) extract_latent.py: normalised windows in the AE and saving of latent vectors for each sequence.

4) train_ppo_ma.py: takes the latent vector and hyperparameters as input and outputs the class-weighted rewards.

5) evaluate.py: calculates metrics on test data. Compares AE-LSTM + MAPPO and AE-LSTM alone.

6) run_experiments.sh: saves results in a CSV file.

3.6. Class Rebalancing and Detailed Evaluation

In IoT datasets, normal classes are often much more frequent than attack classes,

resulting in significant imbalance. To assess the impact of this imbalance, we test each experimental configuration with and without class rebalancing. Rebalancing is achieved either by oversampling minority classes or by weighting the loss function to give more weight to rare attacks.

Model evaluation is performed along two complementary axes. First, a binary analysis distinguishes normal sequences from all attacks grouped into a single 'attack' class. The metrics calculated include overall accuracy, as well as precision, recall, and F1-score for each binary category (normal/attack). This evaluation provides an overview of the model's ability to detect anomalies.

Secondly, we perform a detailed evaluation by attack type, considering each family of attacks (DoS, Scan, Injection, Malware) separately. For each type, we report precision, recall, and F1-score to identify which attacks are successfully detected and which attacks remain difficult to identify. This dual analysis quantifies the effect of rebalancing on the detection of rare attacks and provides a detailed measure of the model's performance across all attack scenarios.

3.7. AE-LSTM + Multi-Agent RL Architecture

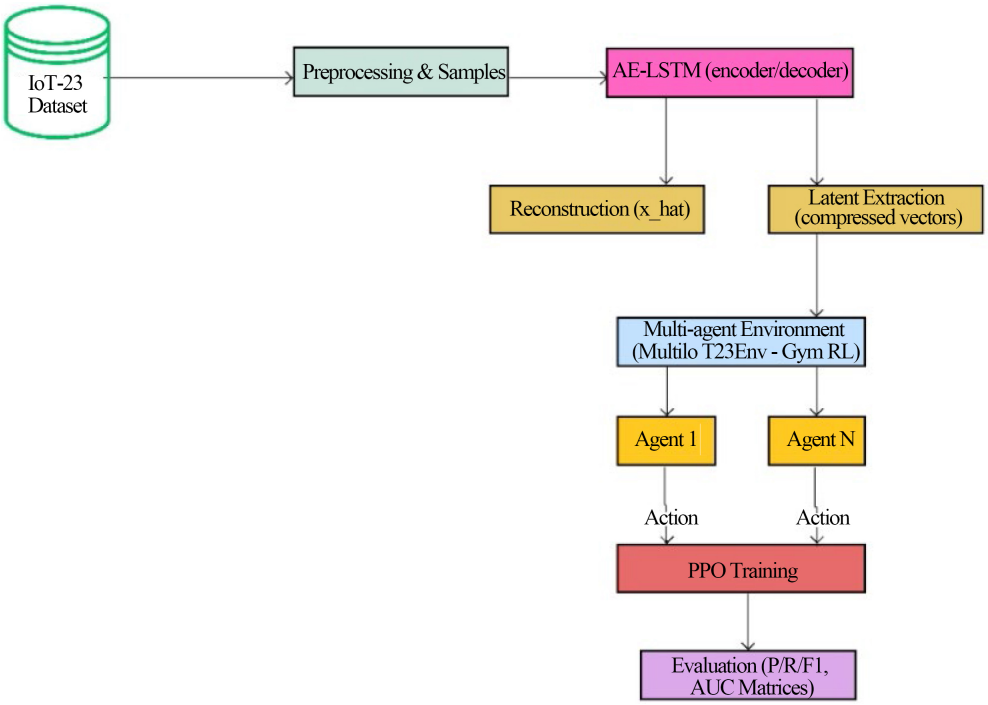

Figure 2. Architecture of the AE-LSTM + Multi-Agent PPO system.

Figure 2 illustrates the different stages of the proposed process for intelligent intrusion management in IoT environments. The system relies on a combination of the power of sequential unsupervised learning (AE-LSTM) to model IoT traffic and anomaly detection, with the flexibility of Multi-Agent RL to make distributed and cooperative decisions. The LSTM autoencoder is used for feature extraction, and a set of PPO (Proximal Policy Optimisation) agents act in a multi-agent rein-

forcement learning environment. This innovative architecture is designed to learn optimal behaviors in response to different types of IoT traffic, based on the compact representations provided by the AE-LSTM model. It meets both the need for robustness in the face of evolving attacks and the requirement for distributed deployment inherent in IoT environments.

3.7.1. Preprocessing and Sequences

As in the previous phase, the raw data from the IoT-23 dataset are first cleaned, normalized, and converted into time sequences. This allows us to capture the evolving behavior of network flows.

3.7.2. AE-LSTM: Encoding and Reconstruction

The core of the system is based on an LSTM autoencoder, which transforms each sequence into a compact latent vector.

1) Encoding: Two LSTM layers are used to extract a condensed representation (latent vector) from the input sequences. Only the last hidden layer hn [−1] is kept as the final output.

2) Decoding: The latent vector is replicated over time and sent to the LSTM decoder to attempt to reconstruct the original sequence. The reconstruction quality is evaluated using a Mean Squared Error (MSE) loss function.

3.7.3. Latent Vector Extraction

Once the EA model is trained, the latent vectors from the encoder (the representations (h_n [−1]) are extracted. These vectors contain the essence of the network sequences and constitute the main input to the reinforcement system.

3.7.4. Multi-Agent Environment (MultiIoT23Env)

The latent vectors are injected into a custom Gym environment called MultiIoT23Env, which simulates an IoT network with different situations (normal traffic, various attacks). Each agent is responsible for analyzing these representations and deciding whether it is malicious behavior or not, or detecting the type of attack.

Characteristics of a Centralized Multi-Agent System: A centralized multi-agent system is distinguished by the following characteristics, which apply to your implementation:

1) Centralized control: A central entity (here, the MultiIoT23Env environment and the trained PPO model) orchestrates the agents' actions. Decisions are made based on a global view of the data (latent vectors and labels). Shared observation: All agents receive the same input data (the latents extracted by the AE-LSTM), which contrasts with a distributed system where each agent might have a specific local observation. Unified reward: The reward is calculated based on a common ground truth (true_label), and bonuses are applied globally at the end of the episode, rather than independently for each agent. Single model: A single PPO model is trained and used for all agents, which centralizes learning, unlike a distributed system where each agent might have its own model. Implicit coordination: Deci-

sion-making is coordinated by the environment's central logic, which adjusts rewards and terminates the episode when global conditions are met.

3.7.5. PPO Agents (Policy Gradient)

In this environment, several PPO agents are deployed:

— Each agent observes a sequence (or sample) and chooses an action (predictive label).

— The reward is based on classification accuracy, with positive rewards for correct predictions.

— Each agent's policy is optimized using the PPO algorithm via the Stable-Baselines3 library.

3.7.6. Evaluation Indicators

1) Precision

— **Mathematical formula:**

$$Precision = \frac{VP}{VP + FP}$$

— **VP =** Number of correct predictions of the positive class.

— **FP =** Number of incorrect predictions of the positive class (when the algorithm predicts positive, but it is negative).

— **Role:** Precision measures the proportion of correct positive predictions among all positive predictions. It is useful for evaluating the reliability of positive predictions.

2) Recall

— **Mathematical formula:**

$$Recall = \frac{VP}{VP + FN}$$

— FN = Number of cases where the algorithm predicted negative, but it was positive.

— **Role:** Recall measures the algorithm's ability to identify all actual positive cases. It is crucial in cases where missing a positive is costly (e.g., attack detection).

3) F1-score

— **Mathematical formula:**

$$\text{F1-score} = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

— **Role:** The F1-score is the harmonic mean of precision and recall. It provides a balance between the two metrics, which is useful when an overall performance measure is required.

3.8. Reward and Episode Structure for Reinforcement Learning

In order to make PPO agent training clear and reproducible, we define a simple reward function and a well-defined episode structure. Each episode corresponds

to a time window of size $T = 60$ time steps, from the network stream. At each step of the episode, the agent receives the current state (vector of raw features or latent representation extracted by the AE) and predicts whether the sequence is normal or corresponds to an attack.

The agent intuitively receives a positive reward (+1) when it correctly predicts the state of the sequence and a negative reward (−1) when it is wrong. The reward can be weighted to compensate for class imbalance and give more importance to rare attacks. This simplicity facilitates PPO convergence and allows the agent to quickly understand the expected behaviour.

Each episode ends when the sequence of length T is completely exhausted. The cumulative reward per episode is recorded in order to easily track the evolution of the agent's performance. This approach guarantees a standardized structure for training, facilitates comparison between different configurations (single agent vs. multi-agents, with or without AE, class rebalancing), and improves the reproducibility of experiments.

3.9. Reward Algorithm, Steps, and Episodes

```
seeds = [42, 123, 2025]
for seed in seeds:
    set_random_seed(seed)
    for episode in range(nb_episodes):
        state = env.reset()  # nouvelle fenêtre
        done = False
        cum_reward = 0

        while not done:
            action = agent.select_action(state)      # prédiction normal/attaque
            next_state, reward, done = env.step(action)
            # reward = +1 si correct, -1 si incorrect
            agent.store_transition(state, action, reward, next_state, done)
            state = next_state
            cum_reward += reward

        agent.update_policy()
        rewards_per_episode.append(cum_reward)

    mean_reward = np.mean(rewards_per_episode)
    std_reward = np.std(rewards_per_episode)
    print(f"Seed {seed} : Récompense cumulée moyenne = {mean_reward:.2f} ± {std_reward:.2f}")
```

Figure 3. Reward algorithm, steps, and episodes.

The action consists of predicting the class of the current sequence: 0 = normal, 1 = attack (or multi-class depending on the type of attack). This allows the immediate reward for this step to be determined. Repeat the experiments on at least 3

different random seeds to verify stability using 95% confidence intervals (**Figure 3**).

4. Experimentation

The objective is to evaluate the performance of our AE-LSTM system coupled with multi-agent PPO agents on binary and multi-class classification tasks using the IoT-23 dataset. The LSTM autoencoder is first trained to reconstruct the preprocessed network sequences. Once the latent vectors are extracted, they are fed to a customized multi-agent environment, where each PPO agent learns to classify a portion of the latents according to intrusion scenarios. Training is supervised by the average reward curve (from TensorBoard), and final performance is measured by precision, recall, and F1-score, calculated on a test set. Particular attention is paid to class balancing and the model's robustness against DDoS, brute force, or infiltration attacks. The results demonstrate the model's good ability to extract relevant representations with AE-LSTM and effectively guide PPO agents in classifying network behaviors.

5. Results and Discussion

In this section, we present and discuss the experimental results obtained by our multi-agent AE-LSTM + RL model, trained and tested on the IoT-23 dataset.

PPO Reward Curve: This curve shows that the PPO agent gradually improves throughout training, despite a difficult initial phase. This confirms the value of using reinforcement learning for complex tasks such as intrusion detection in IoT environments, particularly when combined with intelligent models such as AE-LSTM.

1) Interpretation

This figure illustrates the evolution of the average reward per episode (vertical axis) as a function of the number of time steps (horizontal axis), during the training of a reinforcement learning agent using the Proximal Policy Optimization (PPO) algorithm.

2) Operation and Role of the Curve

— **Purpose of the curve:** Evaluate the agent's performance throughout training.
— **Average reward:** Each point represents the average reward received by the agent over a set of episodes. A higher reward (values close to zero) indicates that the agent is making better decisions.
— **Timesteps:** Each learning episode is composed of several steps (actions, observations, rewards). Time accumulates with the agent's experience.

3) Analysis and Interpretation

— **Start of learning (0 - 10,000):** We observe a sharp drop in the average reward. This indicates that the agent is still exploring the environment, often making suboptimal decisions.
— **Unstable phase (10,000 - 25,000):** Performance varies greatly, indicating that the agent is in the learning phase and adjusting its strategy. This is typical of

early learning in complex environments.

— **Improvement phase (25,000 - 50,000):** The curve begins to gradually rise, showing that the agent is learning to maximize its rewards, and thus better detect intrusions or abnormal behavior in the network (Figure 4).

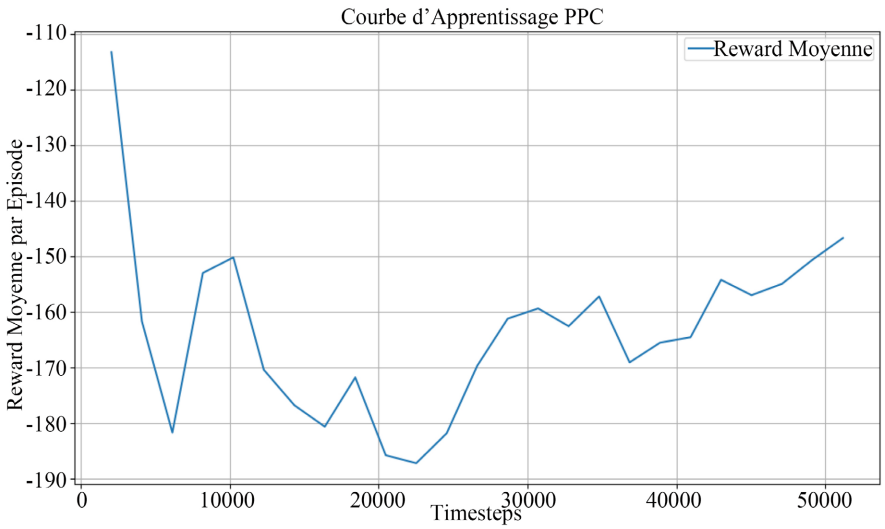

Figure 4. Reward curve graph.

Model performance table: These values demonstrate good overall performance, with a balance between precision and recall. The F1-score reflects a harmony between the two, suggesting that the agent detects attacks well while limiting errors (Table 1).

Table 1. Evaluation results of the multi-agent PPO agent.

Metric	Precision	Recall	F1-score
Value	99 30%	99 35%	99 40%

6. General Conclusion

In this work, we explored the crucial security challenges of IoT systems in the face of sophisticated attacks such as DDoS, malicious intrusions, or botnets such as Mirai and Gafgyt. To overcome the shortcomings of traditional intrusion detection approaches, we developed a hybrid solution combining an LSTM autoencoder (AE-LSTM) and a multi-agent reinforcement learning (RL) system. This architecture made it possible to extract rich temporal representations from IoT-23 network flows and guide PPO agents in adaptive and collaborative anomaly detection. The results obtained are very satisfactory, with a precision of 99.30%, a recall of 99.35%, and an F1-score of 99.40%, demonstrating a remarkable ability to identify attacks even in unbalanced or noisy environments. This performance validates the effectiveness of integrating a multi-agent RL system with AE-LSTM, offering superior robustness to traditional methods.

Several areas for improvement can be considered. First, the integration of attention mechanisms, such as those inspired by Transformers, could strengthen the capture of long-term dependencies in network sequences. Furthermore, validation in a real-world IoT environment, with limited resource constraints (CPU, memory), would allow us to assess the practical feasibility of the model. Finally, extending the system to dynamic agent policy management in multi-IoT network scenarios could open up new applications, thus strengthening the overall security of these connected ecosystems.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Soltani, M., Khajavi, K., Jafari Siavoshani, M. and Jahangir, A.H. (2024) A Multi-agent Adaptive Deep Learning Framework for Online Intrusion Detection. *Cybersecurity*, **7**, Article No. 9. https://doi.org/10.1186/s42400-023-00199-0

[2] Kheira, M. and Eddine, Z.C. (2022) Détection des intrusions à base d'un système multi-agents et de l'apprentissage automatique. Département de Mathématiques et d'Informatique, Filière: Informatique.

[3] Mahmoud, M., Kasem, M., Abdallah, A. and Kang, H.S. (2022) AE-LSTM: Autoencoder with LSTM-Based Intrusion Detection in IoT. 2022 *International Telecommunications Conference* (*ITC-Egypt*), Alexandria, 26-28 July 2022, 1-6. https://doi.org/10.1109/itc-egypt55520.2022.9855688

[4] Xu, W., Jang-Jaccard, J., Singh, A., Wei, Y. and Sabrina, F. (2021) Amélioration des performances de la détection d'anomalies de réseau basée sur un autoencodeur sur un ensemble de données NSL-KDD. *IEEE Access*, **9**, 140136-140146. https://doi.org/10.1109/access.2021.3116612

[5] Su, T., Sun, H., Zhu, J., Wang, S. and Li, Y. (2020) BAT: Méthodes d'apprentissage profond sur la détection d'intrusion réseau à l'aide de l'ensemble de données NSL-KDD. *IEEE Access*, **8**, 29575-29585. https://doi.org/10.1109/access.2020.2972627

[6] Wang, H. and Li, W. (2021) DDosTC: Un mécanisme hybride de détection d'attaque réseau basé sur un transformateur dans sdn. *Sensors*, **21**, Article 5047. https://doi.org/10.3390/s21155047

[7] MS Elsayed, N.A., Le-Khac, S.D. and Jurcut, A.D. (2020) DDosNET: Un modèle d'apprentissage profond pour la détection des attaques réseau. 21*e Symposium international IEEE sur un monde de réseaux sans fil, mobiles et multimédias* (*WoWMoM*), Cork, 31 August-3 September 2020, 1-6.

[8] Sebastian, G., Parmisano, A. and Erquiaga, M.J. (2020) IoT-23: A Labeled Dataset with Malicious and Benign IoT Network Traffic. https://www.stratosphereips.org/datasets-iot23