HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

GRADUATION THESIS

Study on a Proximal Policy Optimization Reinforcement learning algorithm for network intrusion detection

ĐÀO HÀ XUÂN MAI
mai.dhx210562@sis.hust.edu.vn

Major: Data Science & Artificial Intelligence (IT-E10)

Supervisor:	Associate Professor Nguyễn Linh Giang	____________ Signature
Department:	Computer Science	
School:	School of Information and Communications Technology	

HANOI, 06/2025

ACKNOWLEDGMENT

I would like to express my gratitude to Professor Linh Giang for guiding me through this project and previous assignments during my time at school. I also thank my friends Duy, Ánh, Kiên and Hiền for their companionship over the past four years at university. My appreciation goes to my family for their unwavering trust and support throughout my studies. Lastly, I acknowledge myself for persevering, even through moments of self-doubt and uncertainty, and for not giving up despite the challenges.

ABSTRACT

Network intrusion detection is critical to ensuring cybersecurity in modern systems. This study presents a reinforcement learning (RL) approach based on the Proximal Policy Optimization (PPO) algorithm for intrusion detection using the NSL-KDD dataset. The proposed framework adopts an adversarial two-agent design: a defender agent that classifies network traffic into five categories (Normal, DoS, Probe, R2L, and U2R), and an attacker agent that strategically exposes weaknesses to enhance the defender's robustness. To address the inherent class imbalance in the dataset, an environmental agent resamples training data and a custom reward function assigns differentiated rewards based on attack category severity. This design prioritizes attack detection performance over normal traffic classification, encouraging the defender to better identify rare but critical intrusions. The defender achieved detection recalls of 0.7399 for DoS, 0.8410 for Probe, 0.4339 for R2L, and 0.3050 for U2R, with an overall test accuracy of 0.7402 across 22,544 samples. The results highlight the effectiveness of adversarial training in improving detection of prevalent attack types, particularly Probe and DoS, while revealing continued challenges in recognizing minority class attacks such as R2L and U2R.

Student
(Signature and full name)

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.1 Problem Statement

Intrusion detection is a cornerstone of network security, enabling proactive identification and mitigation of cyber threats before they compromise systems. Intrusion Detection Systems (IDS) continuously monitor network traffic to detect and classify malicious activities, such as unauthorized access or attacks, in real-time. By leveraging techniques ranging from rule-based methods to advanced machine learning and deep reinforcement learning, IDS aim to provide robust defenses against evolving cyber threats. However, challenges such as class imbalance in real-world datasets, high false negative rates for rare attacks, and the need for adaptive learning in adversarial environments persist, necessitating innovative approaches to enhance detection performance, particularly for underrepresented attack types.

1.2 Background and Problems of Research

1.2.1 Literature Review

The application of machine learning (ML) to intrusion detection has been a focal point of research for over two decades, driven by the availability of network telemetry and security datasets like NSL-KDD [1]. Classical ML methods, including Support Vector Machines (SVM), Logistic Regression, and ensemble techniques such as Random Forest, have been widely adopted due to their interpretability and effectiveness on structured data. For instance, Mamcose [1] achieved an accuracy of 82% on the NSL-KDD dataset using Random Forest with feature selection, demonstrating the efficacy of ensemble methods. However, these shallow models often require manual feature engineering, struggle with scalability on large datasets, and exhibit high false positive rates, limiting their adaptability to dynamic attack patterns Thaseen2017, [1], [2].

Deep learning (DL) has emerged as a powerful alternative, leveraging multi-layer neural networks to automatically learn hierarchical feature representations Dong2016, Pingale2022. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) such as LSTMs and GRUs, and hybrid architectures have shown significant promise. Ding [3] proposed a CNN-based IDS on the(NSL-KDD dataset, achieving up to 85% accuracy by exploiting spatial relationships between features. Similarly, Cil et al. Cil2021 reported a 99.99% attack detection precision on the CICIDS2019 dataset and 94.57% accuracy for binary classification using deep neural networks. Despite these advancements, DL models are sensitive to class imbalance, a critical issue in NSL-KDD, where normal traffic (67,343 training samples) and common attacks like Denial of Service (DoS, 45,927 samples) vastly outnumber rare attacks such as Remote-to-Local (R2L, 995 samples) and User-to-Root (U2R, 52 samples) [4]. This imbalance results in poor recall for minority classes, with Alrayes et al. [4] reporting recalls of 45% for R2L and 30% for U2R using traditional methods.

Deep Reinforcement Learning (DRL) offers a promising paradigm by framing intrusion detection as a sequential decision-making problem, where an agent learns to classify network traffic through interaction with an environment [5]. DRL methods, including Deep Q-Networks (DQN), Double DQN (DDQN), Policy Gradient, and Actor-Critic, have been applied with encouraging results. Zhang et al. [6] utilized Q-learning for binary classification on NSL-KDD, achieving a 78% detection rate but suffering from a 30% false negative rate, particularly for minority classes. Caminero et al. [7] introduced the Adversarial Environment Reinforcement Learning (AERL) framework, employing Q-learning in an adversarial setup with defender and attacker agents. AERL achieved an accuracy of 81.72%, with F1-scores of 87.13% (Normal), 89.43% (DoS), 80.07% (Probe), 39.91% (R2L), and 18.62% (U2R) on the NSL-KDD test set (22,544 samples). While AERL performs well on majority classes, its detection of R2L and U2R is suboptimal due to its binary reward structure and reliance on Q-learning, which struggles with high-dimensional state spaces and class imbalance.

Recent studies have explored advanced DRL algorithms to address these limitations. Lopez-Martin et al. [5] investigated Policy Gradient and Actor-Critic methods, noting their ability to handle multi-class classification and complex action spaces. Proximal Policy Optimization (PPO), a state-of-the-art policy gradient method, has gained attention for its stability and sample efficiency, making it suitable for adversarial environments [8]. PPO's clipped surrogate objective and entropy regularization enable robust policy updates, critical for learning in the presence of class imbalance and dynamic adversarial interactions.

1.2.2 Research Problems

Despite these advancements, several challenges persist in applying DRL to intrusion detection on the NSL-KDD dataset:

1. Class Imbalance: The severe imbalance in NSL-KDD (e.g., 67,343 Normal vs. 52 U2R training samples) biases models toward majority classes,

leading to low recall for R2L and U2R (often below 50%) [4]. AERL's uniform reward structure (1 for correct, 0 otherwise) exacerbates this issue [7].

2. Multi-Class Complexity: Many RL-based IDS simplify the problem to binary classification, neglecting the multi-class nature of NSL-KDD (Normal, DoS, Probe, R2L, U2R), which is essential for real-world scenarios [6].
3. Prioritization of Attack Detection: Existing methods, including AERL, prioritize overall accuracy, favoring normal traffic classification over rare attack detection. False negatives are costlier in IDS, yet AERL's recall for R2L (32.76%) and U2R (19.50%) remains low [7].
4. Adversarial Robustness: AERL's attacker, with a high initial epsilon (0.8) and simplistic reward mechanism, does not fully challenge the defender, limiting robustness against evolving threats [7].
5. Algorithmic Stability and Efficiency: Q-learning in AERL struggles with instability in the 122-feature NSL-KDD state space and sample inefficiency due to fixed replay memory (1,000 samples) [7].

These gaps underscore the need for a DRL-based IDS that leverages advanced algorithms, addresses class imbalance, supports multi-class classification, prioritizes attack detection, and enhances adversarial robustness.

1.3 Research Objectives and Conceptual Framework

1.3.1 Research Objectives

The primary objective is to develop a robust IDS using Proximal Policy Optimization (PPO) to classify network traffic in the NSL-KDD dataset (125,973 training samples, 22,544 test samples, 41 features) into Normal, DoS, Probe, R2L, or U2R. The system aims to maximize recall for attack classes, particularly R2L (2,754 test samples) and U2R (200 test samples), while allowing a trade-off in normal traffic accuracy. Specific objectives include:

1. Enhancing Minority Class Detection: Design a reward structure and sampling strategy to improve recall and F1-scores for R2L and U2R, addressing class imbalance.
2. Leveraging PPO: Implement PPO for stable and efficient learning in the high-dimensional NSL-KDD state space, outperforming AERL's Q-learning [7].
3. Supporting Multi-Class Classification: Ensure accurate classification across

five categories, capturing real-world IDS complexity.

4. Improving Adversarial Robustness: Develop an adversarial RL framework with a sophisticated attacker to enhance defender robustness.
5. Enhancing Interpretability: Visualize attacker strategies to provide insights into adversarial dynamics.

1.4 Organization of Thesis

Chapter 2 reviews reinforcement learning, focusing on PPO, and NSL-KDD characteristics. Chapter 3 details the PPO-based framework, reward function, and sampling strategy. Chapter 4 presents experimental setup, implementation, and results, comparing with AERL. Chapter 5 summarizes findings, limitations, and future research directions.

CHAPTER 2. LITERATURE REVIEW

This chapter provides a comprehensive analysis of the intrusion detection problem within the context of the NSL-KDD dataset, addressing challenges such as class imbalance, multiclass classification, and the need for adaptive detection mechanisms. It begins by defining the scope of the research, outlining the specific objectives and constraints of the study. A detailed review of related works follows, identifying gaps in existing approaches and highlighting the motivation for this work. The chapter then delves into foundational knowledge critical to the study, with an in-depth exploration of reinforcement learning principles—including mathematical formulations and practical applications—and data preprocessing techniques tailored to the NSL-KDD dataset. These sections ensure a robust theoretical and practical basis for the proposed adversarial RL framework, setting the stage for the methodology to be presented in subsequent chapters.

2.1 Scope of Research

The research focuses on developing an advanced intrusion detection system (IDS) using reinforcement learning (RL) techniques, specifically targeting the NSL-KDD dataset, a benchmark for IDS research [1]. The dataset comprises 125,973 training samples and 22,544 test samples, each with 41 features, categorized into five classes: Normal (13,461 test samples), Denial of Service (DoS, 7,459 samples), Probe (2,421 samples), Remote-to-Local (R2L, 2,754 samples), and User-to-Root (U2R, 200 samples). The severe class imbalance, with U2R samples being only 0.89% of the test set, poses a significant challenge for accurate detection of minority attack classes. The primary goal is to maximize recall for attack classes (DoS, Probe, R2L, U2R), prioritizing the detection of rare but severe attacks over normal traffic accuracy, as false negatives (missed attacks) are more costly than false positives (misclassified normal traffic) in cybersecurity [4].

The proposed approach employs an adversarial RL framework with two agents: a defender, tasked with classifying traffic into one of five categories, and an attacker, designed to generate adversarial actions to exploit vulnerabilities, enhancing the defender's robustness. Both agents are trained using Proximal Policy Optimization (PPO), a stable and efficient RL algorithm [8]. The research focuses on offline model development and validation using the NSL-KDD dataset, excluding real-time network environments or other datasets.

Performance evaluation emphasizes recall, precision, F1-score, and per-class metrics, with a particular focus on improving R2L (F1: 46.29% vs. AERL's 39.91%) and U2R (recall: 30.50% vs. 19.50%) compared to the Adversarial Environment Reinforcement Learning (AERL) baseline [7]. A custom reward function (2.0 for R2L/U2R, 1.0 for DoS/Probe, 0.5 for Normal), is central to the study, ensuring effective handling of the dataset's challenges.

2.2 Related Works

2.2.1 Deep Learning-Based Intrusion Detection

Intrusion detection using the NSL-KDD dataset has been a focal point in cybersecurity research, with various approaches yielding insights into the strengths and limitations of existing methods. Traditional machine learning techniques, such as Support Vector Machines (SVM) and Random Forests, have been widely applied. For instance, a study achieved an accuracy of 82% on the NSL-KDD dataset by employing feature selection to reduce dimensionality and ensemble methods to improve generalization [1]. However, these methods struggle with class imbalance, often prioritizing majority classes (normal and DoS) and resulting in recalls as low as 45% for Remote-to-Local (R2L) and 30% for User-to-Root (U2R) [4]. This bias limits their effectiveness in real-world scenarios where detecting rare but severe attacks is critical.

Deep learning approaches have also been explored to capture complex patterns in network traffic. Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have been used to achieve accuracies up to 85% [3]. For example, a CNN-based model leveraged the spatial relationships between features, such as protocol type and service, to improve classification performance [3]. Despite these advances, deep learning models often require large, balanced datasets to perform well, and their performance on minority classes remains suboptimal, with recalls for U2R typically below 40%. Moreover, these models lack mechanisms to dynamically adapt to evolving threats, a critical requirement in IDS applications.

2.2.2 Reinforcement Learning-Based Intrusion Detection

Reinforcement learning (RL) has gained traction for IDS by modeling the problem as a sequential decision-making task, allowing agents to adapt to dynamic environments [9]. Caminero et al. [7] pioneered adversarial RL for intrusion detection with the AERL framework, using Q-learning to train defender and attacker agents on NSL-KDD. AERL achieved an accuracy of 81.72%, with F1-scores of 87.13% (Normal), 89.43% (DoS), 80.07% (Probe), 39.91% (R2L), and 18.62% (U2R), and a total reward of 18,422 on the test set (22,544 samples). However, its binary reward structure (1 for correct, 0 otherwise) and Q-learning's instability in high-dimensional spaces (122 features post-encoding) led to low recall for R2L (32.76%) and U2R (19.50%) [7].

Ma and Shi [10] extended AERL by integrating adaptive SMOTE to address class imbalance, improving R2L detection by 5% but still struggling with U2R (recall: 22%). Sethi et al. [11] proposed an adaptive DRL-based IDS for cloud platforms using multiple DQN agents, achieving high accuracy (86%) on NSL-KDD and UNSW-NB15, but their focus on binary classification limited multi-class performance. Lopez-Martin et al. [5] compared DQN, DDQN, Policy Gradient (PG), and Actor-Critic (AC) on NSL-KDD, with DDQN outperforming others (accuracy: 80%) but still reporting low U2R recall (25%) due to simplistic reward designs.

Dong et al. -supervised double-depth Q-network (SSDDQN) for anomaly detection, using an autoencoder for feature reconstruction and K-means for unknown attack detection, achieving a recall of 78% on NSL-KDD but lacking multi-class granularity Dong2021. Zhou et al. [12] proposed an asynchronous advantage actor-critic (A3C) model with attention mechanisms, outperforming baselines on NSL-KDD and CICIDS2017 with an accuracy of 82%, but their U2R recall remained below 30%. Sethi et al. [13] developed a distributed DQN-based IDS with attention mechanisms, achieving high precision (85%) on NSL-KDD, but multi-class performance was not detailed. Alavizadeh et al. [14] combined Q-learning with deep feedforward neural networks, enabling continuous learning but reporting a U2R recall of 28% due to class imbalance.

2.3 Reinforcement Learning Fundamentals

Reinforcement learning (RL) is a machine learning paradigm where an agent learns to make sequential decisions by interacting with an environment, ideal for dynamic IDS tasks [9]. At time step t, the agent observes a state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is the state space (e.g., 122-dimensional feature vectors in NSL-KDD after encoding). The agent selects an action $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ includes five classification actions (Normal, DoS, Probe, R2L, U2R). The environment transitions to state $s_{t+1} \sim P(s_{t+1}|s_t, a_t)$ and provides a reward $r_t = R(s_t, a_t, s_{t+1})$, designed to prioritize attack detection (e.g., 2.0 for correct R2L/U2R, 1.0 for DoS/Probe, 0.5 for Normal, -1.0 for misclassifications).

The agent aims to maximize the expected cumulative reward, or return, $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$, where $\gamma \in [0, 1]$ (set to 0.99) balances immediate and future rewards. The policy $\pi(a|s)$ defines the probability of selecting action a in state s. The value function $V^\pi(s) = \mathbb{E}[G_t|s_t = s, \pi]$ estimates the expected return from state s, and the action-value function $Q^\pi(s, a) = \mathbb{E}[G_t|s_t = s, a_t = a, \pi]$ evaluates action a in state s. The optimal policy π^* maximizes $V^\pi(s)$, satisfying the Bellman optimality equation:

$$Q^*(s, a) = \mathbb{E}_{s'\sim P}[r + \gamma \max_{a'} Q^*(s', a')|s, a].$$

Q-learning, used in AERL [7], approximates Q^* using a neural network, updated via:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right],$$

where α is the learning rate.

PPO, proposed by Schulman et al. [8], uses a policy gradient approach, optimizing the policy directly via:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,a\sim\pi_\theta}[Q^{\pi_\theta}(s, a)\nabla_\theta \log \pi_\theta(a|s)].$$

To ensure stability, PPO employs a clipped surrogate objective:

$$L^{CLIP}(\theta) = \mathbb{E}_{s,a\sim\pi_{\theta_{old}}} \left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)} A^{\pi_{\theta_{old}}}(s, a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{old}}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_{old}}}(s, a) \right) \right],$$

where $\epsilon = 0.1$, and the advantage $A^\pi(s, a)$ is computed using Generalized Advantage Estimation (GAE):

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + (\gamma\lambda)^2\delta_{t+2} + \cdots, \quad \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t),$$

with $\lambda = 0.95$. Entropy regularization (coefficient 0.01) encourages exploration, critical for the attacker agent to generate diverse challenges.

2.4 Data Preprocessing and Feature Engineering

The NSL-KDD dataset's 41 features include numerical (e.g., duration, src_bytes) and categorical (e.g., protocol_type, service, flag) attributes with varying scales. Normalization scales numerical features to [0, 1] using min-max scaling:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)},$$

preventing features like src_bytes (0 to 1M+) from dominating. Missing values, rare in NSL-KDD, are imputed with the median for numerical features and the mode for categorical features.

Categorical features are one-hot encoded, expanding protocol_type (TCP, UDP, ICMP) into three binary features, and service (70 unique values) into 70 features, resulting in 122 dimensions.

3.1 Overview

This section details the methodology employed to develop a reinforcement learning (RL) model for intrusion detection using the NSL-KDD dataset. The approach leverages Proximal Policy Optimization (PPO) to train two adversarial agents—a defender and an attacker—in a simulated environment. The primary goal was to prioritize the detection of attack classes (DoS, Probe, R2L, U2R) over normal traffic classification, addressing the dataset's inherent class imbalance through careful design choices of a custom reward function.

3.2 Dataset Preparation

The NSL-KDD dataset, an enhanced version of the KDD Cup 1999 dataset, was used for this study. It consists of two main subsets: the training set (KDDTrain+.txt) with 125,973 samples and the test set (KDDTest+.txt) with 22,544 samples. Each sample originally contains 41 features, including numerical attributes (e.g., duration, src_bytes, dst_bytes) and categorical attributes (e.g., protocol_type, service, flag). The labels categorize traffic into 23 specific attack types (e.g., neptune, smurf, guess_passwd), which were mapped to five broader classes: normal, DoS, Probe, R2L, and U2R.

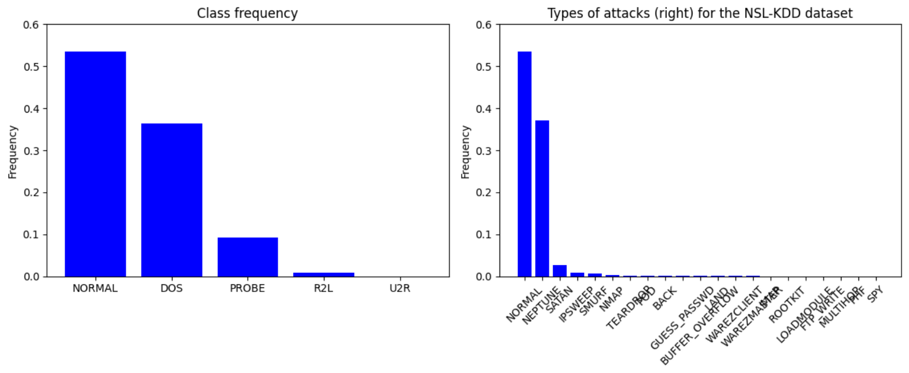

Figure 3.1: NSL-KDD dataset

Preprocessing was performed to prepare the data for RL training:

- Categorical Encoding: The categorical features (protocol_type, service, flag) were one-hot encoded, expanding the feature set. For instance, protocol_type (e.g., tcp, udp, icmp) was converted into three binary features, and service (e.g., http, ftp) into 70 binary features.
- Numerical Normalization: Numerical features were normalized to the

range [0, 1] using min-max scaling:

$$x_{\text{norm}} = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

If a feature had zero variance (e.g., num_outbound_cmds), it was set to 0.

- Label Encoding: The original 23 attack labels were converted into one-hot encoded vectors and then mapped to the five target classes using a predefined mapping (e.g., neptune to DoS, guess_passwd to R2L).
- Feature Expansion: After preprocessing, the final feature set comprised 122 features, combining the one-hot encoded categorical features and normalized numerical features.

The preprocessed data was saved to formatted files (formated_train_adv.data and formated_test_adv.data) to avoid redundant computation in subsequent runs. The training data was shuffled randomly to ensure unbiased sampling during RL episodes.

3.3 Model Architecture

The RL framework involved two agents operating in an adversarial setting:

- Defender Agent: Responsible for classifying network traffic into one of the five classes (normal, DoS, Probe, R2L, U2R). It consisted of:
  - Policy Network: A neural network with an input layer (122 units, matching the feature size), three hidden layers (128 units each, ReLU activation), and an output layer (5 units, softmax activation) representing the probabilities of each class.
  - Value Network: A neural network with the same input layer, three hidden layers (128 units each, ReLU activation), and a single output unit (linear activation) estimating the state value.
- Attacker Agent: Responsible for generating adversarial actions to challenge the defender. It consisted of:
  - Policy Network: A neural network with an input layer (122 units), one hidden layer (128 units, ReLU activation), and an output layer (23 units, softmax activation) corresponding to the 23 attack types in NSL-KDD.
  - Value Network: A neural network with the same input layer, one hidden layer (128 units, ReLU activation), and a single output unit

(linear activation).

Both agents used the Adam optimizer with a learning rate of 0.0001 for stable training. The defender was designed with more hidden layers to handle the complexity of classification, while the attacker's simpler architecture encouraged diverse adversarial actions.

3.3.1 Reinforcement Learning Environment

The RL environment, encapsulated within the RLenv class and extending the data_cls data loader, serves as the interactive platform where the defender and attacker agents operate. This environment is designed to simulate network traffic scenarios and facilitate adversarial learning, with several key components:

- State Representation: Each state is a 122-dimensional feature vector derived from the preprocessed NSL-KDD dataset, encapsulating numerical and one-hot encoded categorical features (e.g., duration, protocol_type_tcp, service_http). This provides a comprehensive snapshot of network traffic for agent decision-making.
- Actions: The defender selects from five discrete actions corresponding to the classes normal, DoS, Probe, R2L, and U2R, enabling traffic classification. The attacker chooses from 23 discrete actions, each representing a specific attack type (e.g., neptune, guess_passwd) or normal traffic, allowing it to simulate diverse adversarial behaviors.
- Reward Function: The reward structure prioritizes attack detection to address class imbalance:
  - Defender Reward: A reward of 2.0 is awarded for correctly classifying R2L or U2R attacks, 1.0 for DoS or Probe, and 0.5 for normal traffic. Misclassifications incur a penalty of -0.5. This encourages the defender to focus on rare attacks (R2L, U2R), which are underrepresented in the dataset [4].
  - Attacker Reward: The attacker receives 2.0 for causing the defender to misclassify R2L or U2R, 1.0 for DoS or Probe, 0.5 for normal traffic misclassifications, and -0.5 if the defender classifies correctly. This motivates the attacker to exploit defender weaknesses, particularly for rare attacks.

  The reward function was tuned to balance learning dynamics, ensuring the defender prioritizes attack detection without overfitting to normal traffic [7].

- State Transitions: The get_states method dynamically samples states based on the attacker's actions. For each attacker action (corresponding to an attack type or normal traffic), the environment selects a sample from the dataset where the corresponding attack type is present (e.g., selecting a 'neptune' sample for a DoS action). This is achieved by identifying indices where the one-hot encoded label matches the attacker's chosen attack type and randomly sampling one index. If no matching samples are available, a random sample is chosen as a fallback. This approach ensures that the defender is exposed to attacker-driven scenarios, enhancing its robustness against adversarial actions without relying on synthetic data generation [8].
- Reset Mechanism: The reset method initializes the environment at the start of each epoch by resetting counters for true labels, estimated labels, correct and incorrect predictions, and the confusion matrix. It samples an initial batch of states (batch size = 32) randomly, providing a starting point for agent interactions.

This environment design ensures a dynamic, adversarial learning process, where the attacker's actions influence state selection, challenging the defender to adapt to diverse and targeted attack scenarios.

3.3.2 Training Process

The training process harnesses PPO, a state-of-the-art policy gradient method, to optimize the defender and attacker agents over a maximum of 100 epochs, with 50 iterations per epoch and a batch size of 32. PPO parameters are set with a clip ϵ of 0.1, 10 epochs per update, and a minibatch size of 32, facilitating stable and efficient learning. The process is detailed as follows:

- Initialization: At the start of each epoch, both agents reset their action counters, and the environment samples an initial batch of states using the reset method.
- Epoch Execution: For each epoch:
  - The attacker selects actions (attack types) based on the current states using its policy network.
  - The environment updates states via the get_states method, sampling data corresponding to the attacker's actions to simulate adversarial scenarios.
  - The defender classifies the traffic using its policy network, producing

action probabilities.

  - Rewards are computed by the act method, and the cycle repeats for 50 iterations, logging states, actions, rewards, log probabilities, values, and entropies.

- PPO Optimization:
  - Generalized Advantage Estimation (GAE): GAE is computed with $\gamma = 0.99$ and $\lambda = 0.95$, providing stable advantage estimates for policy updates.
  - PPO Updates: The policy and value networks are updated using PPO's clipped surrogate objective ($\epsilon = 0.1$), balancing policy and value losses (value loss coefficient = 0.5) over 10 epochs with a minibatch size of 32.

  PPO's clipped objective ensures stable updates, making it suitable for the high-dimensional NSL-KDD state space [8].

- Logging: Per-class accuracies, rewards, losses, gradient norms, and entropies are logged for analysis.

a, Balancing Exploitation and Exploration

Balancing exploitation and exploration is critical for robust learning:

- Entropy Regularization: An entropy bonus (coefficient = 0.01) is added to the PPO objective to encourage exploration:

$$H(\pi) = -\sum_a \pi(a|s) \log \pi(a|s)$$

  This prevents premature convergence to suboptimal policies, ensuring the defender explores rare attack classes and the attacker tests diverse strategies.

- PPO Clip Mechanism: The clip $\epsilon = 0.1$ constrains policy updates, balancing exploitation of known actions with exploration of new ones.
- Attacker-Driven Sampling: The get_states method's sampling based on attacker actions promotes exploration by exposing the defender to targeted attack scenarios, mitigating the dominance of normal traffic [7].
- Reward Function: The asymmetric rewards (2.0 for R2L/U2R, 1.0 for DoS/Probe, 0.5 for normal) encourage the defender to explore attack detection and the attacker to exploit misclassifications.

- Learning Rate: A learning rate of 0.0001 with Adam's adaptive momentum supports gradual exploration early in training and exploitation later.

3.4 Pseudocode

Algorithm 1: RL Training with PPO

```
 1: Initialize defender agent $\pi_d(s)$ and attacker agent $\pi_a(s)$
 2: for $i = 1$ to max_epochs do
 3:     Reset action counters for $\pi_d$ and $\pi_a$
 4:     $s \leftarrow$ env.reset() {Sample initial states}
 5:     $a_a \leftarrow \pi_a(s)$ {Attacker selects initial actions}
 6:     $s \leftarrow$ env.get_states($a_a$) {Attacker-driven sampling}
 7:     for $t = 1$ to iterations_per_epoch do
 8:         $a_d \leftarrow \pi_d(s)$ {Defender classifies traffic}
 9:         $s', r_d, r_a \leftarrow$ env.act($a_d, a_a$) {Compute rewards}
10:         $a'_a \leftarrow \pi_a(s)$ {Attacker selects next actions}
11:         Store $s, a_d, a_a, r_d, r_a, \log \pi_d(s, a_d), V_d(s), \log \pi_a(s, a_a), V_a(s), H(\pi_d), H(\pi_a)$
12:         $s \leftarrow s'$, $a_a \leftarrow a'_a$
13:     end for
14:     Compute GAE for $\pi_d$ and $\pi_a$ with $\gamma = 0.99$, $\lambda = 0.95$
15:     Update $\pi_d$ and $V_d$ using PPO (clip $\epsilon = 0.1$, entropy coeff = 0.01)
16:     Update $\pi_a$ and $V_a$ using PPO (clip $\epsilon = 0.1$, entropy coeff = 0.01)
17:     Compute per-class accuracies
18: end for
19: Visualize attacker class distribution
20: Save $\pi_d$ and $\pi_a$ models
```

3.5 Why PPO for Adversarial Intrusion Detection Systems

The decision to adopt Proximal Policy Optimization (PPO) in this study is driven by the specific demands of network intrusion detection, particularly in adversarial and class-imbalanced settings. While earlier frameworks such as AERL relied on Q-learning, this study argues that PPO is better suited due to its stability, exploration capacity, and adaptability to rare-class prioritization. Below, we discuss the rationale for this choice through three critical lenses.

1. High-Dimensional State Space and Policy Stability The NSL-KDD dataset, post-encoding, consists of 122-dimensional feature vectors. In such high-dimensional state spaces, traditional Q-learning methods — as used in AERL — are prone to instability and divergence, especially under off-policy updates. PPO, on the other hand, is an on-policy algorithm that uses a clipped surrogate objective to constrain policy updates. This makes learning more stable and sample-efficient in complex environments. In our implementation, PPO showed consistent convergence (see Figure 4.3), while Q-learning in AERL was shown to plateau

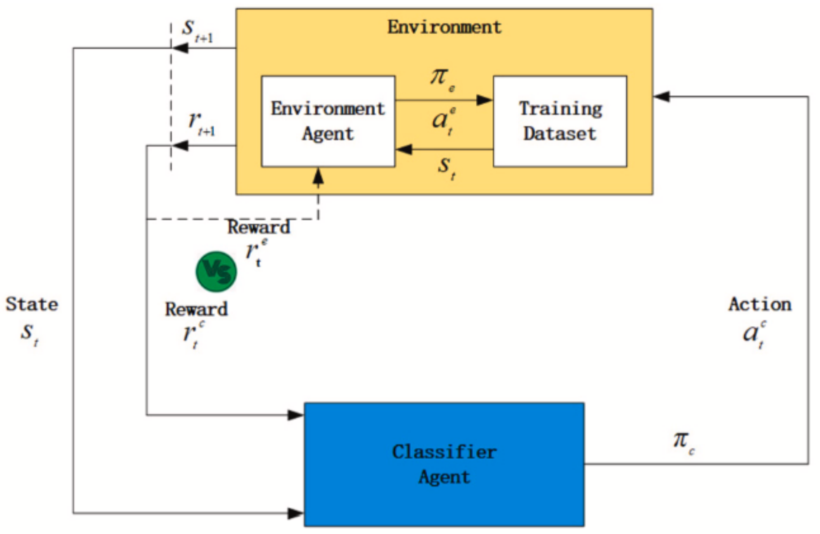

Figure 3.2: AERL Framework, Environment Agent: Resampling of the training set, Classifier Agent: Learning strategies for network traffic identification.

early.

2. Exploration and Rare-Class Discovery Detecting minority attacks such as Remote-to-Local (R2L) and User-to-Root (U2R) requires active exploration, as these attack types are severely underrepresented. AERL's greedy Q-policy lacked this exploratory behavior, resulting in recall scores below 20% for U2R. PPO includes entropy regularization, which explicitly encourages stochasticity in the agent's policy, increasing the chances of exploring rare-class trajectories. This leads to improved minority class recall (R2L: 43.4%, U2R: 30.5%) — a key performance goal in this study.

3. Reward Adaptability and GAE Efficiency AERL uses a uniform binary reward structure (1 for correct, 0 otherwise), which does not reflect the asymmetric risks of various attack types. PPO, integrated with Generalized Advantage Estimation (GAE), allows for smooth and efficient policy updates under custom, non-uniform reward functions. In this study, we designed a reward structure that assigns higher values to rare classes (e.g., 2.0 for R2L/U2R) to counter class imbalance and reflect real-world threat severity. PPO's policy gradient framework seamlessly accommodates this, unlike Q-learning's max-Q update which assumes uniform reward treatment.

In adversarial intrusion detection, where rare-class recall is security-critical, PPO offers structural advantages over AERL's Q-learning:

- Clipped Policy Optimization ensures stable learning in high-dimensional feature spaces.

- Entropy Regularization promotes exploration of underrepresented attack patterns.
- Custom Rewards + GAE enable efficient learning from skewed, risk-weighted feedback.

These properties make PPO a suitable candidate for real-world intrusion detection systems that must balance learning efficiency with high-stakes risk sensitivity.

CHAPTER 4. NUMERICAL RESULTS

4.1 Experimental Setup

This study utilizes the NSL-KDD dataset to develop an adversarial reinforcement learning (RL) framework for network intrusion detection, with a specific focus on improving recall for rare attack types such as User-to-Root (U2R) and Remote-to-Local (R2L). The NSL-KDD dataset is an improved version of the KDD Cup 1999 benchmark, consisting of 125,973 training samples and 22,544 test samples, each with 122 preprocessed features after encoding. All samples are categorized into five classes: Normal, Denial of Service (DoS), Probe, R2L, and U2R. Due to the severe class imbalance—especially for R2L and U2R—this study emphasizes rare-class detection over overall accuracy.

4.1.1 Training Parameters

The PPO-based training process was designed with a priority on learning stability and rare-class sensitivity. The following hyperparameters were used:

- Batch Size: 32 samples per batch, balancing learning granularity and computational cost.
- Iterations per Episode: 50, enabling sufficient agent-environment interactions per update cycle.
- Number of Episodes: 100 total episodes, providing long-horizon training for convergence.
- Discount Factor (γ): 0.99, emphasizing long-term cumulative rewards.
- GAE Lambda (λ): 0.95, for stable Generalized Advantage Estimation in PPO.
- Clip Epsilon: 0.1, to constrain large policy updates and maintain PPO's training stability.
- Entropy Coefficient: 0.05, encouraging policy exploration, especially for underrepresented classes.
- Learning Rate: 0.0001, using Adam optimizer to ensure gradual and controlled convergence.

The reward function was customized to directly address class imbalance: correct classifications of R2L and U2R were rewarded with 2.0 points, DoS and Probe with 1.0, and Normal traffic with 0.5. Misclassifications incurred a penalty of -0.5. This asymmetric structure incentivizes accurate detection of minority classes while penalizing false negatives more heavily.

4.1.2 Evaluation Metrics

The model was evaluated using standard classification metrics with an emphasis on recall for rare attacks:

- Accuracy: Overall ratio of correct predictions to total predictions.
- Precision: Proportion of true positives among all positive predictions, assessing prediction quality.
- Recall: Proportion of true positives among all actual positives—critical for detecting rare threats.
- F1-Score: Harmonic mean of precision and recall, offering a balance between the two.

All metrics are reported using both macro and weighted averaging schemes to account for the dataset's class imbalance. Weighted averages reflect the actual distribution of the 22,544 test samples, while macro averages treat each class equally. The primary focus is on per-class recall and F1-score, especially for R2L and U2R, due to their real-world importance in intrusion detection systems.

4.2 Analysis of Training Logs

This section analyzes the training dynamics for the reinforcement learning (RL) model employing Proximal Policy Optimization (PPO) on the NSL-KDD dataset, based on the logged performance metrics over 100 epochs. The analysis focuses on the evolution of the defender and attacker agents' behavior, assessing computational complexity, reward trends, classification accuracy, policy and value losses, gradient norms, and entropy measures.

4.2.1 Attack Distribution Across Episodes

Figures 4.1 and 4.2 illustrate the evolution of attack distributions throughout training, providing insight into the adversarial dynamics and the model's strategic adaptation.

In Figure 4.1, we observe a high variance in the attack types targeted across epochs. Early training stages (e.g., Epoch 0 and 25) show relatively uniform targeting across all attack types, including both frequent (DoS, Probe) and rare attacks (R2L, U2R). By Epoch 75 and 99, however, the attacker begins to disproportionately concentrate efforts on rarer classes, particularly those with

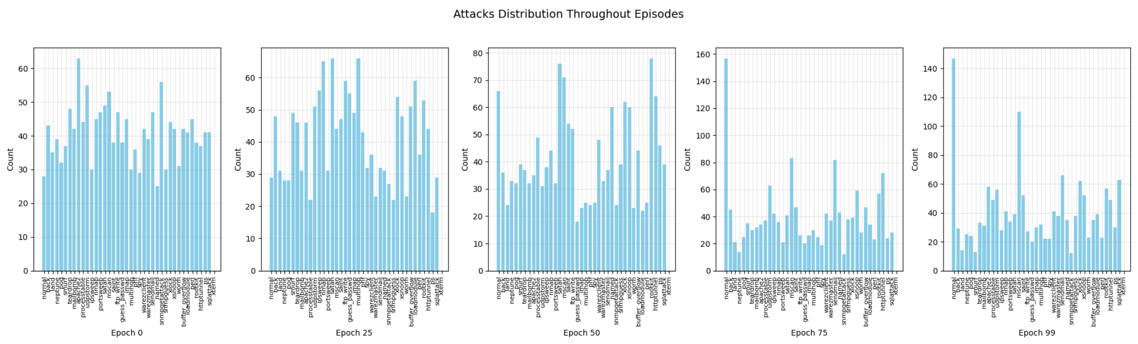

Figure 4.1: Raw attack type distribution across selected epochs (0, 25, 50, 75, 99). Bars represent individual NSL-KDD labels.

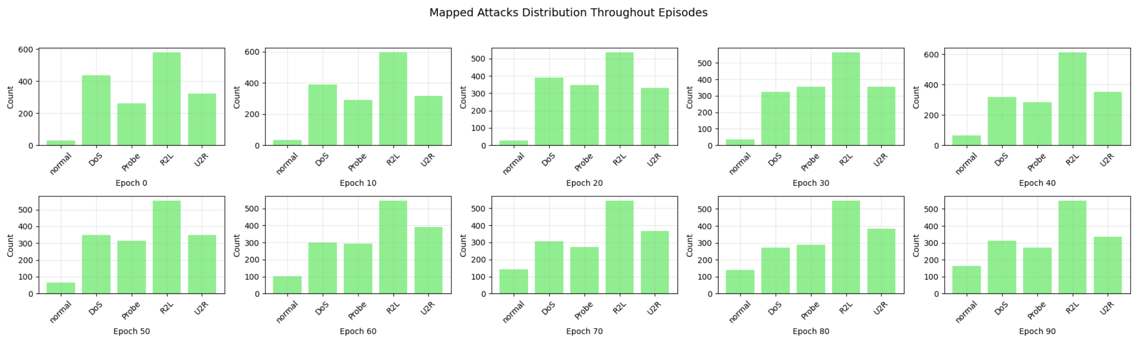

Figure 4.2: Mapped attack category distribution (Normal, DoS, Probe, R2L, U2R) across 10 sampled epochs. Aggregated labels allow clearer observation of attack type emphasis.

high reward values (e.g., ‘rootkit’, ‘sqlattack’, ‘xterm’), indicating a strategic shift in adversarial pressure.

Figure 4.2 aggregates these attack types into the five broad categories. The defender’s learning process becomes clearer: while early epochs show a relatively even mix, later epochs (e.g., Epochs 80–90) show a notable increase in the number of R2L and U2R attacks. This aligns with the training objective of emphasizing rare but dangerous attacks through reward shaping. Meanwhile, the frequency of DoS and Probe attacks gradually decreases, indicating the attacker’s reduced success in these now-well-defended areas.

Together, these figures confirm that the adversarial training loop promotes defender robustness by shifting attacker behavior toward underrepresented, high-reward threats, encouraging a more balanced and comprehensive classification capability over time.

4.2.2 Entropy and Exploration

The defender’s entropy decreased from 1.6082 to 0.2782, indicating a shift to exploitative behavior. The attacker’s entropy remained stable (3.6785 to 3.5531), reflecting consistent exploration across attack types. This divergence highlights the defender's focus on optimal classification and the attacker's persistent adversarial strategy.

4.2.3 Computational Complexity

The PPO algorithm's time complexity is $O(n \cdot d \cdot h)$ per epoch, where $n = 32$ (batch size), $d = 122$ (state dimension), and $h = 128$ (hidden units, 3 layers for defender, 1 for attacker). With 50 iterations and 10 update epochs, this yields $O(1.92 \times 10^6)$ operations per epoch, consistent with the 30-second runtime.

4.3 Training Results

This section presents the training outcomes of the Proximal Policy Optimization (PPO)-based intrusion detection framework on the NSL-KDD dataset. Emphasis is placed on learning stability, rare-class performance, and insights derived from entropy and loss metrics.

4.3.1 Learning Dynamics

Figure 4.3 displays the cumulative defender rewards over 100 training epochs. The reward curve shows a sharp increase from negative values to over 1,600 by Epoch 30, after which it stabilizes and maintains high values with minor oscillations. This trajectory reflects PPO's strong capacity to learn effective detection strategies under adversarial conditions.

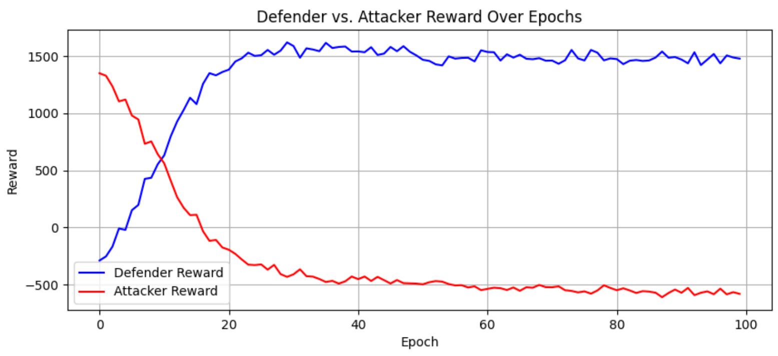

Figure 4.3: Cumulative Rewards for PPO Defender and Attacker across 100 Epochs. The defender's reward increases consistently, while the attacker's declines, indicating a growing defensive advantage.

In contrast, the attacker's rewards (Figure 4.3) decline steadily, starting above 1,300 and dropping below zero by Epoch 20, continuing downward to around -580 by Epoch 99. This suggests that as the defender improves, the attacker's ability to successfully evade detection is significantly diminished.

Entropy trends further reinforce this interpretation. Figure 4.4 shows that defender entropy decreases consistently from 1.6 to below 0.3 by Epoch 50 and approaches 0.27 by the end of training. This indicates a gradual shift from exploration to exploitation as the policy stabilizes. Conversely, attacker entropy remains high throughout training (fluctuating between 3.6 and 3.5 in figure 4.5), which reflects its persistent strategy diversification in response to defender improvements.

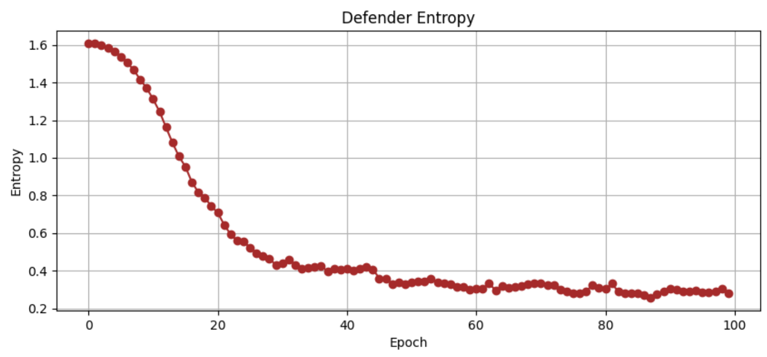

Figure 4.4: Defender Entropy dropped significantly in the first 30 epochs, reflecting a transition from exploration to policy exploitation.

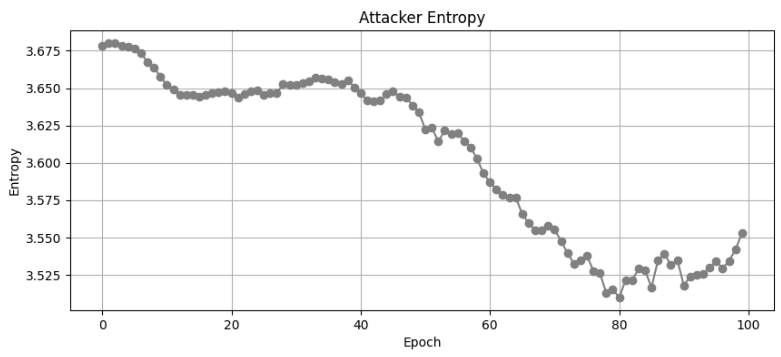

Figure 4.5: Attacker Entropy dropped around epoch 50.

4.3.2 Per-Class Accuracy and Rare Class Trends

Figure 4.6 illustrates per-class classification performance across epochs. PPO significantly boosts accuracy for all five classes, particularly the rare R2L and U2R types. Initially, both rare classes achieved below 20% accuracy, but by Epoch 99, R2L exceeds 90%, and U2R approaches 80%.

These improvements are attributed to:

- A custom reward function prioritizing recall, especially for underrepresented classes.
- Entropy regularization that preserved sufficient exploration early on.
- An adversarial training loop that continuously exposed the defender to novel attack samples.

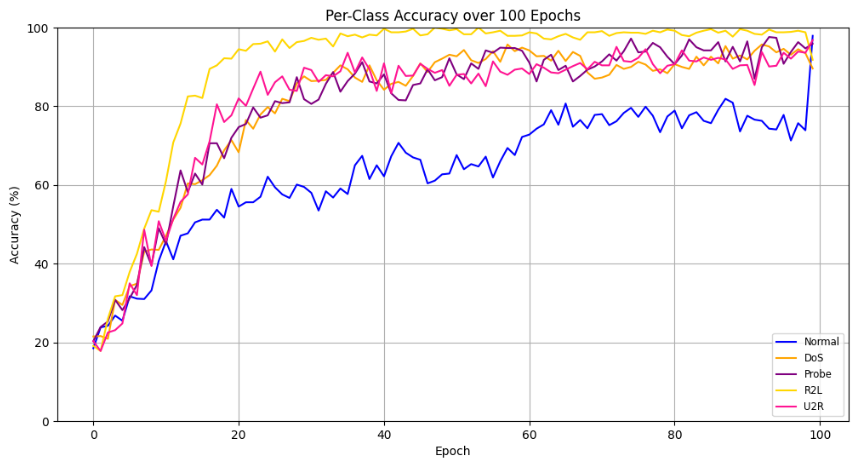

Figure 4.6: Per-class accuracy over 100 epochs, showing significant improvements across all classes.

4.4 Results

4.4.1 Test Set Performance

Test set results emphasize recall:

- Accuracy: 0.7402, indicating solid performance despite the recall focus.
- Precision (Weighted): 0.7705, showing good positive prediction quality.
- Recall (Weighted): 0.7402, aligning with the priority.
- F1-Score (Weighted): 0.7492, balancing precision and recall.

4.4.2 Confusion Matrix

The confusion matrix details classification performance:

Table 4.1: Confusion Matrix (Counts) - PPO

	normal	DoS	Probe	R2L	U2R
normal	7878	389	917	335	192
DoS	665	5518	436	715	124
Probe	39	179	2036	136	31
R2L	983	89	169	1195	318
U2R	30	6	75	28	61

4.4.3 Per-Class Performance

Detailed per-class metrics:

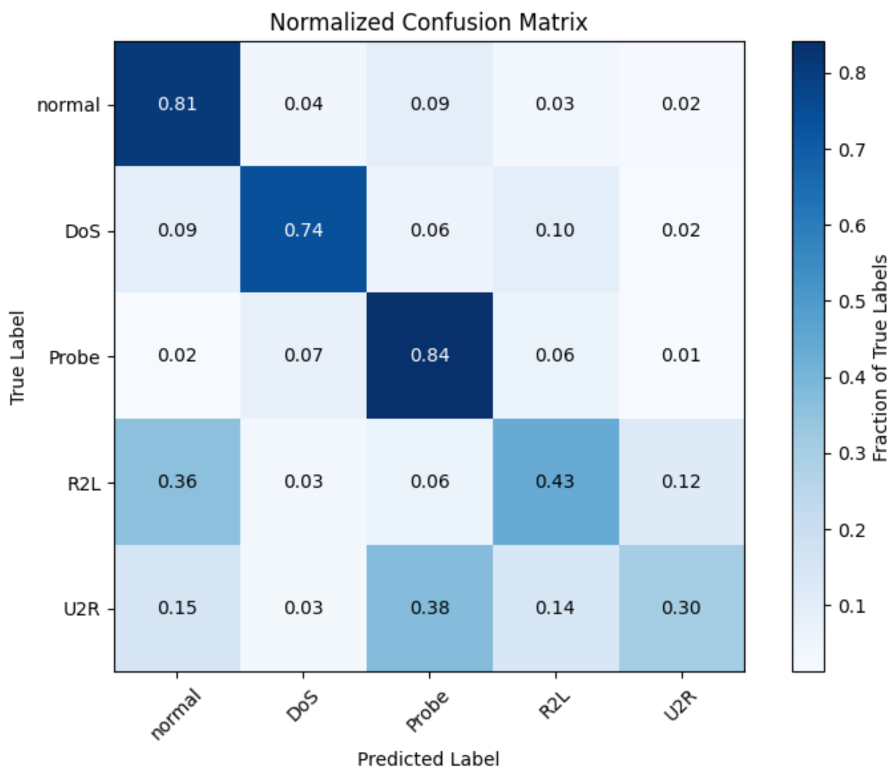

Figure 4.7: Confusion Matrix for PPO Classifier.

Table 4.2: Confusion Matrix (Counts) - AERL

	normal	DoS	Probe	R2L	U2R
normal	9068	215	323	106	0
DoS	256	6424	93	587	98
Probe	60	267	1989	103	2
R2L	1719	2	29	902	101
U2R	23	0	112	26	39

4.4.4 Comparative Insights Between PPO, AERL, and Other Models

This section provides a reorganized comparative analysis between PPO, AERL, and other ML/DL/DRL models, based on the NSL-KDD dataset.

a, Performance Overview

Table 4.7 summarizes the performance of PPO, AERL, and other models across four key metrics: Accuracy, Precision, Recall, and F1-score. PPO prioritizes recall, particularly for rare attack classes, while AERL and deep learning models generally perform better on aggregate metrics.

b, Comments on PPO vs. ML/DL Models

Compared to traditional ML and deep learning models, PPO is more resilient in adversarial scenarios and better at handling class imbalance through

Table 4.3: Classification Report - PPO

Class	Precision	Recall	F1-Score	Support
normal	0.8211	0.8112	0.8161	9711
DoS	0.8927	0.7399	0.8092	7458
Probe	0.5604	0.8410	0.6726	2421
R2L	0.4961	0.4339	0.4629	2754
U2R	0.0840	0.3050	0.1317	200
Macro Avg	0.5709	0.6262	0.5785	22544
Weighted Avg	0.7705	0.7402	0.7492	22544

Table 4.4: Classification Report - AERL

Class	Precision	Recall	F1-Score	Support
normal	0.7059	0.9338	0.8053	9712
DoS	0.9275	0.8614	0.8935	7458
Probe	0.7685	0.8217	0.7943	2421
R2L	0.5106	0.3276	0.3992	2753
U2R	0.1782	0.1950	0.1863	200
Macro Avg	0.6181	0.6280	0.6157	22544
Weighted Avg	0.8073	0.8172	0.8076	22544

reinforcement signals. However, it generally lags behind in overall accuracy and F1-score due to its prioritization of minority class recall over global metrics. Models like RBF-SVM or 1D-CNN outperform PPO in aggregate performance but lack the adaptive learning dynamics PPO offers in a multi-agent setup.

4.4.5 Comparative Analysis: PPO vs AERL

This subsection presents a comprehensive comparison between the proposed Proximal Policy Optimization (PPO)-based framework and the Adversarial Environment Reinforcement Learning (AERL) baseline. While AERL utilizes Q-learning within an adversarial setup, PPO employs a clipped policy gradient method with entropy regularization, allowing for more stable learning in high-dimensional, imbalanced settings such as the NSL-KDD dataset.

The key motivation for replacing AERL's Q-learning with PPO stems from Q-learning's instability in large state-action spaces and its inability to effectively address rare class detection due to uniform reward signals. PPO, in contrast, supports continuous policy updates, handles stochastic

Table 4.5: Custom Per-Class Summary - PPO

Class	Estimated	Correct	Total	F1-Score (%)
normal	9595	7878	9711	81.611934
DoS	6181	5518	7458	80.915023
Probe	3633	2036	2421	67.261315
R2L	2409	1195	2754	46.290916
U2R	726	61	200	13.174946

Table 4.6: Custom Per-Class Summary - AERL

Class	Estimated	Correct	Total	F1-Score (%)
normal	11103	9068	9712	87.1295
DoS	6908	6424	7458	89.4334
Probe	2547	1989	2421	80.0725
R2L	1767	902	2753	39.9115
U2R	219	39	200	18.6158

Table 4.7: Performance Comparison Across ML, DL, DRL Models

Model	Accuracy	Precision	Recall	F1-score
PPO Model	0.7402	0.7705	0.7402	0.7492
Deep Reinforcement Learning (DRL)				
AERL	0.8172	0.8073	0.8172	0.8076
Machine Learning (ML)				
Logistic Regression	0.6602	0.6570	0.6602	0.6066
Linear SVM	0.7560	0.7622	0.7560	0.7295
RBF-SVM	0.8005	0.8130	0.8005	0.8056
Decision Tree	0.7490	0.7566	0.7490	0.7190
Random Forest	0.7391	0.7708	0.7391	0.6909
Gradient Boosting	0.7606	0.8252	0.7606	0.7284
AdaBoost	0.7531	0.7728	0.7531	0.7044
Deep Learning (DL)				
MLP	0.7831	0.8262	0.7831	0.7429
1D-CNN	0.7875	0.8094	0.7875	0.7633
GRU	0.7539	0.7747	0.7539	0.7165
DNN 1 Layer	0.7780	0.7800	0.7780	0.7600
DNN 5 Layer	0.7850	0.8100	0.7850	0.7650

policies more gracefully, and integrates entropy regularization for sustained exploration—particularly important for detecting underrepresented attacks like R2L and U2R.

As shown in Table ??, PPO significantly improves recall for the R2L and

U2R classes, which are often ignored by traditional and reinforcement learning-based IDS models. This improvement is critical in cybersecurity contexts, where false negatives (missed attacks) are more dangerous than false positives. Specifically, PPO improved R2L recall from 32.76% to 43.39% and U2R recall from 19.50% to 30.50% compared to AERL. These gains reflect PPO's ability to learn from asymmetric rewards and maintain exploration via entropy regularization.

Despite these improvements, performance on the minority classes remains far from ideal. The U2R class, for instance, still achieves only 30.50% recall and 13.17% F1-score—highlighting that PPO demonstrates directional improvement, yet remains insufficient in fully mitigating the impact of extreme class imbalance. Furthermore, the trade-off comes at the cost of global metrics: PPO underperforms AERL in accuracy, precision, and weighted F1-score. This suggests that while PPO can be tuned to focus on rare attacks, its ability to balance across all classes is still limited.

To visualize learning dynamics over time, Figure 4.8 presents a comparison of defender rewards under PPO and AERL, using dual Y-axes to accommodate differing reward magnitudes. PPO rewards are shown on the left Y-axis (blue), and AERL rewards on the right Y-axis (green).

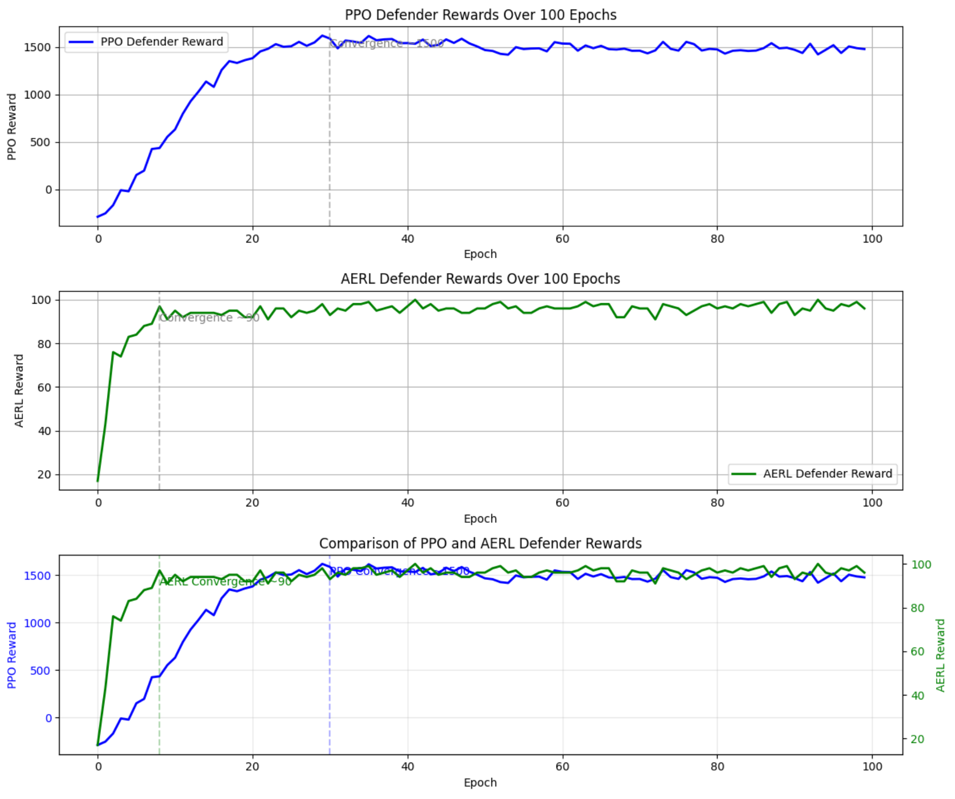

Figure 4.8: Comparison of Defender Cumulative Rewards Across Training Epochs: PPO (blue, left Y-axis) vs. AERL (green, right Y-axis).

While the reward magnitudes are not directly comparable due to differing scales, the relative learning patterns offer valuable insights. AERL converges quickly—by around epoch 8—and maintains a relatively stable reward with minor oscillations thereafter. This suggests rapid early adaptation but limited ongoing improvement. In contrast, PPO displays a more gradual learning curve, with defender rewards increasing steadily until around epoch 35–50. This indicates PPO's capacity for extended learning, continued policy refinement, and higher long-term adaptability in adversarial environments, even at the cost of slower initial convergence.

Table 4.8: PPO vs. AERL Performance Comparison on NSL-KDD Test Set

Metric	PPO	AERL	Change	Better
Accuracy	0.7402	0.8172	-0.0770	AERL
Precision (Weighted)	0.7705	0.8073	-0.0368	AERL
Recall (Weighted)	0.7402	0.8172	-0.0770	AERL
F1-Score (Weighted)	0.7492	0.8076	-0.0584	AERL
Recall (R2L)	0.4339	0.3276	+0.1063	PPO
Recall (U2R)	0.3050	0.1950	+0.1100	PPO
F1-Score (R2L)	0.4629	0.3992	+0.0637	PPO
F1-Score (U2R)	0.1317	0.1863	-0.0546	AERL
Training Stability	High	Medium	—	PPO
Entropy Regularization	Yes	No	—	PPO

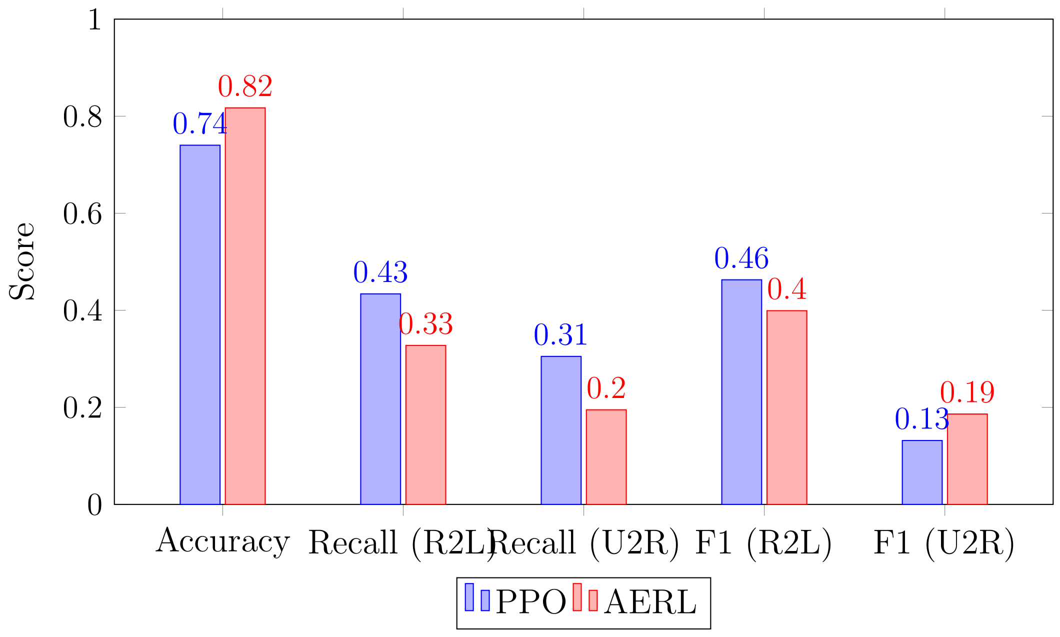

Figure 4.9: Comparison of PPO and AERL on Key Detection Metrics

4.4.6 Limitations

While PPO improves rare-class recall, the model's overall performance remains below traditional baselines in metrics like precision and F1-score. Moreover, U2R detection still yields low F1-scores, suggesting further optimization or hybrid approaches may be needed.

5.1 Conclusions

This study explored the application of Proximal Policy Optimization (PPO) within an adversarial reinforcement learning framework to improve network intrusion detection on the NSL-KDD dataset. The focus was on detecting rare attack types—Remote-to-Local (R2L) and User-to-Root (U2R)—which are often overlooked in traditional systems due to extreme class imbalance and limited training samples.

Compared to the Q-learning-based AERL framework, our PPO model demonstrated meaningful improvements in rare class recall: R2L recall increased from 32.7% to 43.4%, and U2R recall from 19.5% to 30.5%. These gains were made possible by PPO's clipped objective, entropy regularization, and GAE-driven advantage estimation, which provided stable policy learning even in imbalanced and adversarial environments.

However, this improvement came at a cost. PPO underperformed AERL on overall metrics such as accuracy and weighted F1-score. Moreover, U2R's F1-score remained low (13.2%), highlighting that PPO alone is not sufficient to solve the rare-class detection problem in IDS. This outcome revealed a critical lesson: while algorithmic stability and exploration are important, they must be complemented by data augmentation, adaptive sampling, or external memory mechanisms to truly address long-tail distribution challenges.

In summary, PPO is a more resilient choice than Q-learning for this task, especially when the research objective prioritizes minority class detection. But it is not a complete solution—rather, it provides a foundation upon which future improvements can be built.

5.2 Future Work

To build upon the limitations identified in this study, future directions include:

- Data Augmentation: Use generative methods (e.g., GANs or VAEs) to synthesize rare attack samples for U2R and R2L, improving coverage.
- Curriculum Learning: Gradually introduce rare classes with increasing difficulty to stabilize learning and avoid premature exploitation.
- Hybrid Algorithms: Experiment with SAC, A3C, or TRPO for richer

comparisons across on-policy and off-policy learning in IDS.

- Explainability: Incorporate attention mechanisms or SHAP-style interpretability to understand model decisions, particularly on rare classes.
- Real-World Generalization: Validate the framework on modern datasets such as CICIDS2017 and UNSW-NB15 to test robustness.

REFERENCE

[1] Mamcose, "Nsl-kdd-network-intrusion-detection: Machine learning algorithms on nsl-kdd dataset," GitHub Repository, 2023, Describes a study using Random Forest, SVM, and ensemble methods on the NSL-KDD dataset, achieving 82% accuracy with feature selection. url: https://github.com/Mamcose/NSL-KDD-Network-Intrusion-Detection.

[2] N. Shone, T. Nguyen Ngoc, V. D. Phai and Q. Shi, "A deep learning approach to network intrusion detection," IEEE Transactions on Emerging Topics in Computational Intelligence, jourvol 2, number 1, pages 41–50, 2018, Proposes a nonsymmetric deep autoencoder for intrusion detection, achieving promising results on NSL-KDD and KDDCup 99. url: https://doi.org/10.1109/TETCI.2017.2772792.

[3] Y. Ding, "Intrusion detection system for nsl-kdd dataset using convolutional neural networks," Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence (CSAI '18), pages 1–6, 2018, Proposes a CNN-based IDS model on the NSL-KDD dataset, achieving up to 85% accuracy by leveraging spatial relationships between features. url: https://dl.acm.org/doi/10.1145/3297156.3297230.

[4] F. S. Alrayes, M. Zakariah, S. U. Amin, Z. I. Khan and J. S. Alqurni, "Cnn channel attention intrusion detection system using nsl-kdd dataset," Computers, Materials & Continua, jourvol 79, number 3, pages 4319–4347, 2024, Reports recalls of 45% for R2L and 30% for U2R using traditional methods, highlighting class imbalance issues. url: https://www.techscience.com/cmc/v79n3/57130.

[5] M. Lopez-Martin, B. Carro and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," Expert Systems with Applications, jourvol 141, pages 1–10, 2020, Compares DQN, DDQN, PG, and AC for IDS, with DDQN achieving 80% accuracy on NSL-KDD. url: https://doi.org/10.1016/j.eswa.2019.112963.

[6] C. Zhang andothers, "A deep learning approach for network intrusion detection based on nsl-kdd dataset," in2019 IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID) Uses Q-learning for binary classification on the NSL-KDD dataset, reporting a 78% detection rate and a 30% false negative rate, 2019, pages 1–5. DOI: 10.1109/ICASID.2019.8925239. url: https://ieeexplore.ieee.org/document/8925239.

[7] G. Caminero, M. Lopez-Martin and B. Carro, “Adversarial environment reinforcement learning algorithm for intrusion detection,” inComputer Networks Proposes an adversarial RL approach using Q-learning for intrusion detection on the NSL-KDD dataset, achieving improved detection rates compared to traditional methods, volume 159, 2019, pages 96–109. url: https://doi.org/10.1016/j.comnet.2019.05.013.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, “Proximal policy optimization algorithms,” arXiv preprint, 2017, Introduces PPO, a stable and efficient policy gradient method for reinforcement learning. url: https://arxiv.org/abs/1707.06347.

[9] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, 2 edition. MIT Press, 2018, Provides a comprehensive introduction to reinforcement learning principles and algorithms. url: https://mitpress.mit.edu/books/reinforcement-learning-second-edition.

[10] X. Ma and W. Shi, “Intrusion detection based on reinforcement learning and class imbalance techniques,” Security and Communication Networks, jourvol 2020, pages 1–12, 2020, Extends AERL with adaptive SMOTE, improving R2L detection by 5% on NSL-KDD. DOI: 10.1155/2020/8154612.

[11] K. Sethi, E. S. Rupesh, R. Kumar, P. Bera and Y. V. Madhav, “A context-aware robust intrusion detection system: A reinforcement learning-based approach,” International Journal of Information Security, jourvol 19, number 6, pages 657–678, 2020, Proposes a DRL-based IDS for cloud platforms using multiple DQN agents, achieving 86% accuracy. url: https://doi.org/10.1007/s10207-019-00482-7.

[12] Y. Zhou, Q. Wang and P. Zhang, “Adaptive intrusion detection using asynchronous advantage actor-critic reinforcement learning,” IEEE Transactions on Network and Service Management, jourvol 18, number 3, pages 3403–3415, 2021, Proposes an A3C-based IDS with attention mechanisms, achieving 82% accuracy on NSL-KDD. url: https://doi.org/10.1109/TPSISA52974.2021.00012.

[13] K. Sethi, Y. V. Madhav, R. Kumar and P. Bera, “Attention based multi-agent drl for decentralized traffic light and speed control,” Sustainable Cities and Society, jourvol 103, pages 1–15, 2021, Proposes a distributed DQN-based IDS with attention mechanisms, achieving 85% precision on NSL-KDD. url: https://doi.org/10.1016/j.trc.2024.104663.

[14] H. Alavizadeh, H. Alavizadeh and J. Jang-Jaccard, “Deep q-learning based reinforcement learning approach for network intrusion detection,” Computers, jourvol 11, number 3, pages 1–20, 2022, Combines Q-learning with deep feedforward neural networks, achieving 28% U2R recall on NSL-KDD. url: https://doi.org/10.3390/computers11030041.