An Offset-Attention Boosted Double Deep Q-Network for Network Intrusion Detection

Cuixia Li, Shuxiao Wang, Zhenghao Yang, Jizhe Zhao, Shuyan Zhang*
School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou, China
*Corresponding author: syzhang@zzu.edu.cn

***Abstract*—Network Intrusion Detection (NID) serves as a critical defense against increasingly sophisticated cyber threats. While Deep Reinforcement Learning (DRL) has been widely adopted in NID research, existing methods often suffer from unsatisfactory detection accuracy. Due to the dominance of benign traffic in real-world networks, standard DRL agents tend to exhibit a bias towards the majority class, resulting in sub-optimal overall performance and poor sensitivity to fine-grained, minority attack families. To address this, an Offset-Attention Enhanced Double Deep Q-Network (OA-DDQN) is proposed. A novel Offset-Attention (OA) mechanism is introduced to replace standard global aggregation with a subtractive interaction. By calculating the offset between input features and their attention-weighted representations, this process acts as a learnable Laplacian operator, explicitly sharpening the distinction between benign traffic and subtle malicious activities. Besides, the Double Deep Q-Network (DDQN) is employed to provide precise value estimation, thereby mitigating the bias towards majority classes and ensuring robust decision-making. Extensive comparative experiments on the NSL-KDD benchmark validate the effectiveness of the proposed approach, where OA-DDQN outperforms state-of-the-art baselines and achieves the highest overall detection accuracy.**

***Index Terms*—Network Intrusion Detection, Deep Reinforcement Learning, Double Deep Q-Network, Feature Extraction, Offset-Attention.**

I. Introduction

Network Intrusion Detection (NID) is a critical component of cybersecurity, aiming to identify malicious activities from network traffic in real time.

Traditional NID methods mainly rely on misuse detection or anomaly detection [1]. However, signature-based methods generalize poorly to unknown attacks, while anomaly-based methods often depend on handcrafted features and suffer from high false alarm rates. Deep Reinforcement Learning (DRL) offers a promising alternative by enabling adaptive decision-making through interaction with the environment.

However, the performance of DRL-based NID still depends heavily on feature representation [2]. Although Self-Attention (SA) can capture global correlations [3], its weighted aggregation may smooth subtle local anomalies, which is unfavorable for detecting minority attack families such as U2R in NSL-KDD.

To address these limitations, the Offset-Attention Double Deep Q-Network (OA-DDQN) is proposed. A novel Offset-Attention (OA) mechanism is introduced to explicitly calculate the difference between input features and their attention-weighted representations, this operation can simulates a learnable Laplacian operator, sharpening the feature space to highlight subtle anomalies. Subsequently, the Double Deep Q-Network (DDQN) architecture is adopted as the decision-making backbone, serving to precisely map these enhanced feature representations into optimal detection actions.

The main contributions of this paper are summarized as follows:

- We propose OA-DDQN, a DRL-based NID framework that alleviates the feature smoothing issue of standard self-attention.
- We adopt a practical preprocessing pipeline on NSL-KDD, including official train/test partitioning, hybrid resampling [4], [5], and correlation-based feature selection.
- We conduct comprehensive experiments and ablation studies to validate the effectiveness of OA-DDQN against representative baselines.

The source code is publicly available[1].

II. Related Work

Existing NID solutions can be broadly grouped into three categories: traditional deep learning methods, DRL-based methods, and attention-based feature representation methods.

A. Traditional and Deep Learning Methods for NID

Early NID systems mainly relied on signature matching and classical machine learning. Signature-based methods are effective for known attacks but fail on unseen threats, whereas shallow anomaly-detection models such as SVM and RF depend heavily on handcrafted features and often suffer from false positives. Deep learning models, such as CNNs and RNNs, improve automatic feature extraction, but their supervised nature limits adaptability to evolving attack patterns without retraining [6].

B. Deep Reinforcement Learning for NID

To address the adaptability issues of supervised models, DRL has been introduced into NID. Unlike static classification, DRL enables agents to learn optimal detection strategies through continuous interaction with the network environment [7], [8]. For instance, Deep Deterministic Policy Gradient (DDPG) has been applied to handle continuous action spaces in traffic control, while Proximal Policy Optimization (PPO) has been utilized to improve training stability and policy convergence.

[1] https://anonymous.4open.science/r/OA-DDQN-D299

Despite these successes, value-based methods remain the most direct approach for discrete anomaly classification. However, the standard Deep Q-Network (DQN) [9] inherently suffers from the Q-value overestimation problem, where the maximization step in the target calculation leads to inflated value estimates. This bias can cause significant policy oscillation and instability during the training process, hindering the reliability of the NID. To mitigate this, the DDQN [10] was proposed to decouple action selection from evaluation. Its robustness in modern security tasks has been validated in recent studies [11], [12], serving as a solid foundation for our framework.

C. Attention Mechanisms for Traffic Feature Representation

Effective feature representation is critical for the decision-making of DRL agents. While standard DL layers can extract basic features, they often struggle to capture long-range dependencies in complex traffic flows. To solve this, the SA mechanism, originally proposed for Natural Language Processing [13], has been integrated into NID frameworks. The SA calculates global correlation scores, enabling the model to focus on the most discriminative segments of the traffic payload, which has proven effective in improving Detection Rates (DR) [14].

However, the SA mechanism calculates weights based on global similarities, which tends to produce a smoothing effect on the feature map. In the context of high-dimensional network traffic, this global focus may inadvertently overlook subtle local differences or fine-grained relative positional shifts between normal and malicious packets. Inspired by the success of offset-based methods in point cloud processing [15], we adapt this paradigm to network traffic analysis. Unlike standard SA which focuses on global similarities, our approach introduces a subtractive interaction to specifically enhance the agent's sensitivity to fine-grained local anomalies, filling the gap in capturing subtle attack signatures.

III. Methodology

This section presents the proposed OA-DDQN framework, including data preprocessing, OA-based feature enhancement, RL interaction, and DDQN training.

A. Data Preprocessing

We utilize the NSL-KDD dataset [16]. Unlike many studies that randomly split the dataset, which may lead to information leakage and biased results, we strictly adhere to the official dataset partitioning. The model is trained using the designated training set and evaluated on the separate, unseen testing set. This rigorous setting ensures a fair comparison with existing benchmarks and tests the model's generalization capability against unknown traffic patterns.

To address the challenges of data quality, we first tackle the severe class imbalance inherent in network traffic. Benign traffic significantly accounts for the majority, and direct training on such data biases the model towards the majority class, leading to poor detection of minority attacks. To mitigate this, we employ a hybrid resampling strategy combining the SMOTE [4] and Tomek Links [5].

First, we apply the SMOTE to oversample the minority classes by interpolating between existing samples, thereby balancing the class distribution. Subsequently, we utilize Tomek Links to clean the overlapping data. A Tomek link is defined as a pair of samples from different classes that are each other's nearest neighbors. By removing these ambiguous pairs, we eliminate the noise introduced by SMOTE and sharpen the decision boundaries between classes. This hybrid strategy improves the classifier's robustness by providing a balanced and distinct training set.

The dataset NSL-KDD originally contains 41 traffic features. Many of these are redundant or highly correlated, which increases computational overhead and the risk of overfitting. Given that network traffic features often exhibit non-linear relationships, we adopt the Spearman's Rank Correlation Coefficient for feature selection. The correlation matrix is computed based on rank variables, and features with an absolute correlation coefficient greater than 0.95 are removed. This process effectively reduced the feature dimension while retaining the most discriminative information.

B. Feature Enhancement via OA-DDQN Architecture

This subsection details the internal neural mechanics that specifically implement the Feature Enhancement. To process the input state into actionable Q-values, we design a hierarchical network architecture consisting of three functional modules: Feature Tokenization,OA Encoder, and the Q-Value Prediction Head.

Feature Tokenization: To bridge the gap between data preprocessing and the neural network, we first employ a tokenization module. Let $x \in \mathbb{R}^D$ denote the preprocessed traffic feature vector from Stage 1. We map each scalar feature to a high-dimensional token to obtain the initial input H_0:

$$H_0 = \mathit{Linear}(x) + E_{id} \tag{1}$$

where $\mathit{Linear}(\cdot)$ projects the scalar value into the feature embedding space, and E_{id} denotes the learnable field embedding vectors suitable for attention operations.

Offset-Attention Encoder: This module is the core innovation responsible for Feature Enhancement. As formulated in (2), the representation H_0 is fed into stacked OA layers to capture non-linear correlations:

$$H_l = \mathrm{OA}(H_{l-1}), \quad l = 1, 2, \ldots, L \tag{2}$$

where H_l denotes the refined feature representation output by the l_{th} Offset-Attention layer.

Unlike standard SA which tends to smooth features, our OA mechanism is designed to sharpen them. It first computes the standard global context using (3) and (4):

$$Q = H_{l-1}W_Q, \quad K = H_{l-1}W_K, \quad V = H_{l-1}W_V \tag{3}$$

$$F_{sa} = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_{model}}}\right)V \tag{4}$$

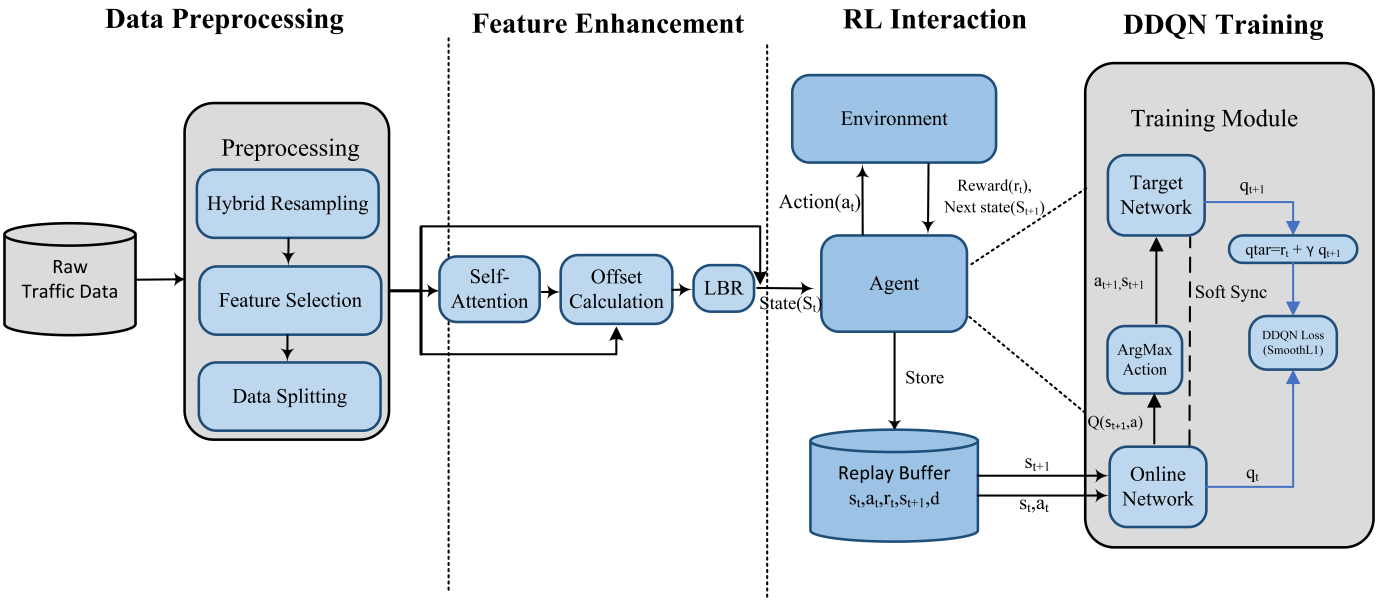

Fig. 1. The overall architecture of the proposed OA-DDQN framework.

where W_Q, W_K, W_V represent the learnable linear projection matrices for the Query (Q), Key (K), and Value (V), respectively. d_{model} is the dimension of the feature embedding, and F_{sa} denotes the generated self-attention feature map that captures global context information.

The critical distinction is visualized in Fig. 2. Instead of using F_{sa} directly, we introduce a subtraction path to calculate the offset between the raw input and the attention features. This offset is then refined by the Linear-BN-ReLU (LBR) block to highlight local anomalies:

$$H_l = \text{LBR}(H_{l-1} - F_{sa}) + H_{l-1} \tag{5}$$

This mathematical operation acts as a learnable Laplacian filter, enhancing the discriminative power of the state representation before it reaches the decision-making agent.

Q-Value Prediction Head: Finally, the enhanced feature H_L is utilized by the Agent for decision-making. The feature map is flattened and processed by a Multi-Layer Perceptron (MLP) to generate the Q-values:

$$Q(s, \cdot) = MLP(Flatten(H_L)) \tag{6}$$

The output vector represents the expected utility of taking each action, serving as the basis for the interaction and subsequent DDQN training.

C. RL Interaction

The interaction between the agent and the network environment is modeled as an MDP tuple $M = (S, A, R, \gamma)$.

- **State Space (S):** The state $s_t \in \mathbb{R}^D$ represents the traffic feature vector at time step t, where D is the dimension of the feature space (e.g., duration, protocol type, flags).
- **Action Space (A):** The action space is discrete. For the binary classification task, $A = \{0, 1\}$, where $a_t = 0$ denotes normal traffic and $a_t = 1$ indicates an attack. For the 5-class task, the action space is expanded to $A = \{0, 1, 2, 3, 4\}$, corresponding to Normal, DoS, Probe, R2L, and U2R, respectively.
- **Reward Function (R):** To prioritize detection Acc, we design a binary reward function. The agent receives a positive reward for correct classifications and zero otherwise:

$$r_t = \begin{cases} 1 & \text{if } a_t = y_t \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

  where y_t is the ground truth label.
- **Discount Factor (γ):** The factor $\gamma \in [0, 1]$ balances the importance of immediate and future rewards.

The primary objective of the agent is to find an optimal policy π^* that maximizes the expected cumulative discounted reward. Formally, the cumulative reward, denoted as R_t at time step t, is defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \tag{8}$$

where r_{t+k} represents the immediate reward received at time step $t + k$.

D. Training Algorithm

The proposed OA-DDQN is trained by minimizing the temporal difference (TD) error between the predicted Q-values and the target Q-values.

To stabilize the learning process, we adopt the Experience Replay mechanism. Instead of updating the network

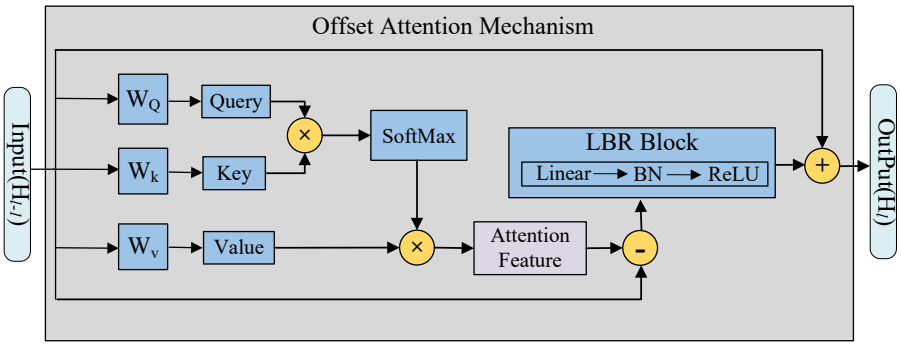

Fig. 2. The detailed architecture of the OA Mechanism.

immediately after each step, the agent stores the transition tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ in a replay buffer D with a fixed capacity. During training, a mini-batch of transitions is uniformly sampled from D to perform gradient descent.

Consequently, the objective is to minimize the expected loss over the sampled distribution D:

$$L(\theta) = E_{(s,a,r,s_t+1)\sim D}\left[SmoothL1\left(y_t - Q(s,a;\theta)\right)\right] \quad (9)$$

To mitigate the overestimation bias, we employ the DDQN strategy, where the target value y_t is computed by decoupling action selection from value evaluation:

$$y_t = r + \gamma Q'\left(s_{t+1}, \arg\max_a Q(s_{t+1}, a; \theta); \theta^-\right) \quad (10)$$

where θ and θ^- represent the parameters of the online network and the target network, respectively. The target network parameters θ^- are periodically updated to match θ.

The detailed training procedure is summarized in Algorithm 1.

Algorithm 1 OA-DDQN Training Algorithm for Network Intrusion Detection

Input: Training dataset $D = \{(x_i, y_i)\}_{i=1}^N$; Batch size m; Max episodes M; η (Learning Rate);γ;Update freq C; Epsilon strategy: $\epsilon_{start}, \epsilon_{end}, \epsilon_{decay}$.
Output: Trained Q-Network parameters θ.
Initialize: Q_θ with random weights; Q_{θ^-} with weights $\theta^- = \theta$; Replay Buffer B (capacity K); Global step $t_{global} \leftarrow 0$; $\epsilon \leftarrow \epsilon_{start}$.

for Episode $e = 1$ to M **do**
  Shuffle training dataset indices
  for each sample (x_t, y_t) in shuffled D **do**
    Observe state $s_t = x_t$
    $t_{global} \leftarrow t_{global} + 1$
    Extract enhanced features from s_t via OA
    Select action a_t via ϵ-greedy policy:
    $$a_t = \begin{cases} \text{random action}, & \text{with probability } \epsilon \\ \arg\max_a Q_\theta(s_t, a), & \text{otherwise} \end{cases}$$
    Execute a_t, observe $r_t = \mathbb{I}(a_t = y_t)$ and s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) into B
    if $|B| \geq m$ **then**
      Sample batch $T = \{(s_j, a_j, r_j, s_{j+1})\}_{j=1}^B \sim B$
      Compute best next action:
      $a'_{max} \leftarrow \arg\max_a Q_\theta(s_{j+1}, a)$
      Compute target value:
      $y_j \leftarrow r_j + \gamma \cdot Q_{\theta^-}(s_{j+1}, a'_{max})$
      Minimize SmoothL1Loss in (9)
      if $t_{global} \pmod C = 0$ **then**
        Update target network: $\theta^- \leftarrow \theta$
      end if
    end if
  end for
  Decay epsilon: $\epsilon \leftarrow \max(\epsilon_{end}, \epsilon \cdot \epsilon_{decay})$
end for

IV. Experiments and Analysis

In this section, we comprehensively evaluate the proposed OA-DDQN model on the NSL-KDD benchmark dataset. The experiments are conducted in two scenarios: binary classification (Normal vs. Attack) and 5-class multi-classification (identifying specific attack families, i.e., Normal, DoS, Probe, R2L, and U2R).

A. Evaluation Metrics

To comprehensively evaluate the performance of OA-DDQN, we utilize standard evaluation metrics derived from the fundamental elements of the confusion matrix. These elements include True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The metrics are defined as follows:

- **Acc:** The ratio of correctly classified samples to the total number of samples.

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

- **Precision :** The proportion of predicted attack flows that are actually attacks.

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

- **Detection Rate(DR):** Also known as Recall, it is the proportion of actual attack flows that are correctly identified.

$$\text{DR} = \frac{TP}{TP + FN} \tag{13}$$

- **False Alarm Rates(FAR):** The proportion of benign traffic incorrectly classified as attacks.

$$\text{FAR} = \frac{FP}{FP + TN} \tag{14}$$

- **F1-Score:** The harmonic mean of Precision and Recall.

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{DR}}{\text{Precision} + \text{DR}} \tag{15}$$

The selection of reported metrics aligns with specific analytical objectives. For binary classification, we prioritize DR and FAR to characterize the operational trade-off between security coverage and maintenance overhead. Conversely, for 5-class classification, we emphasize Precision alongside Accuracy. In imbalanced scenarios, Precision serves as a stricter indicator of discriminative capability, validating the model's ability to identify specific attack patterns without relying on over-prediction.

B. Implementation Details

The detailed specifications of the experimental environment, including the physical computing platform and software versions, are listed in Table I.

TABLE I
SPECIFICATIONS OF THE EXPERIMENTAL ENVIRONMENT

Category	Item	Specification
Hardware	CPU	Intel® Xeon® Gold 6336Y @ 2.40GHz
	GPU	NVIDIA Tesla T4 (16GB Memory)
Software	Language	Python 3.8.1
	Framework	PyTorch 1.9.0
	Acceleration	CUDA 11.4

The proposed OA-DDQN model is implemented using the PyTorch library [17]. We employ the Adam optimizer to update the network weights. The specific hyperparameter configurations used during the training process are summarized in Table II.

TABLE II
HYPERPARAMETER SETTINGS

Parameter	Value
Replay Buffer Size	10,000
Mini-batch Size	128
Learning Rate	5×10^{-4}
Discount Factor (γ)	0.01
Target Update Frequency	100
Initial Epsilon (ϵ_{start})	0.5
Final Epsilon (ϵ_{end})	0.001
Epsilon Decay (ϵ_{decay})	0.75
seed	42

C. Comparison with State-of-the-Art Algorithms

To verify the effectiveness of the proposed OA-DDQN model, we conducted comparative experiments on the NSL-KDD dataset against various baselines, including PPO [18], SAC [19], DQN [9], SA-DDQN [20], MLP [21], and SVM. To ensure a fair comparison and reproducibility, all models were trained and evaluated under the exact same experimental settings. During the evaluation phase, the exploration mechanism was disabled to test the deterministic policy on the unseen testing set. The evaluation covers both binary classification and 5-class attack identification.

As shown in Fig. 3, OA-DDQN achieves the best overall performance in binary classification, reaching about 0.95 in both Acc and F1-score while maintaining the highest DR and the lowest FAR. These results confirm that OA-DDQN is robust in identifying malicious traffic while effectively minimizing false positives.

Table III further reports the class-wise performance in the 5-class setting. OA-DDQN achieves the best overall Acc of 0.8282 and improves minority-class detection, especially for U2R (0.2015), compared with SVM (0.1969) and SA-DDQN (0.1453). Although its DoS performance is slightly lower than that of SA-DDQN, OA-DDQN provides a more balanced trade-off across majority and minority classes.

TABLE III
COMPARISON OF CLASS-WISE PRECISION AND ACC

Model	Class-wise Precision					Acc
	Normal	DoS	Probe	R2L	U2R	
DQN	0.7889	0.9133	0.6222	0.6058	0.1842	0.7889
PPO	0.7146	0.9590	0.5509	0.6231	0.1963	0.7369
SAC	0.7997	0.9293	0.5360	0.7151	0.1066	0.7821
SA-DDQN	0.8333	0.9660	0.6493	0.6030	0.1453	0.8212
MLP	0.6756	0.8628	0.3295	0.8438	0.1620	0.7164
SVM	0.8002	0.9485	0.6326	0.7467	0.1969	0.8040
OA-DDQN	0.8221	0.9350	0.6556	0.7579	0.2015	0.8282

D. Ablation Study

To evaluate the contribution of the OA mechanism, we compare OA-DDQN with the baseline DDQN. As shown in Table IV, OA-DDQN improves Acc from 0.9179 to 0.9470 and F1-score from 0.9276 to 0.9536, while reducing FAR from 0.0910 to 0.0597. These results confirm the effectiveness of OA in improving both detection accuracy and robustness.

TABLE IV
ABLATION STUDY: OVERALL PERFORMANCE ON TEST SET

Model	Acc	F1-Score	DR	FAR
DDQN	0.9179	0.9276	0.9246	0.0910
OA-DDQN	0.9470	0.9536	0.9512	0.0597

Table V further shows that OA-DDQN consistently improves minority-class detection, especially for Probe, R2L, and U2R, although a slight trade-off is observed on DoS. Overall, OA narrows the performance gap between majority and minority classes.

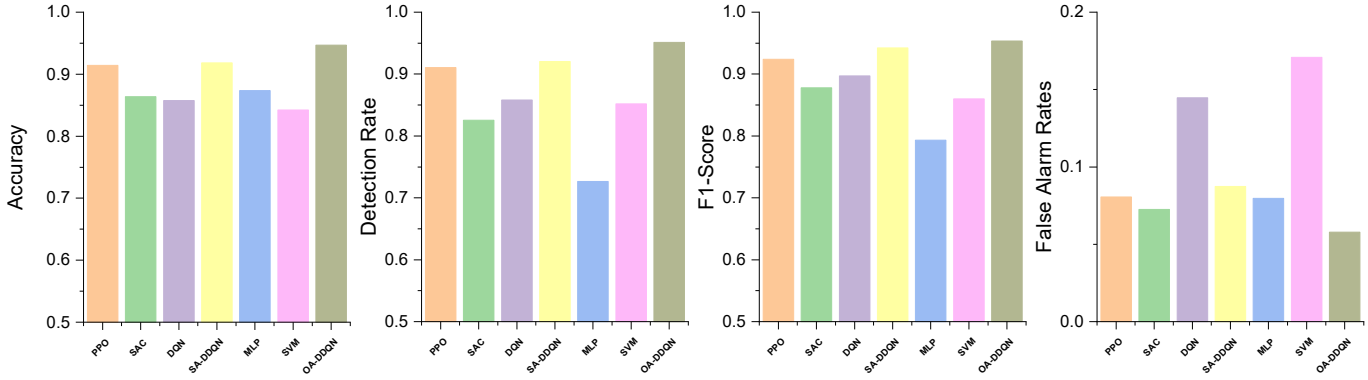

Fig. 3. Performance comparison of different models in terms of Acc, DR, F1-Score, and FAR. The proposed OA-DDQN demonstrates superior performance across all metrics.

TABLE V
ABLATION STUDY: IMPACT OF OFFSET-ATTENTION ON CLASS-WISE PRECISION AND ACC

Model	Class-wise Precision					Acc
	Normal	DoS	Probe	R2L	U2R	
DDQN	0.8069	0.9745	0.5525	0.6372	0.1767	0.8019
OA-DDQN	0.8221	0.9350	0.6556	0.7579	0.2015	0.8282

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed OA-DDQN for network intrusion detection. By introducing an offset-attention mechanism, the proposed framework alleviates the feature smoothing issue of standard self-attention and improves the detection of fine-grained anomalies. Experiments on NSL-KDD demonstrate that OA-DDQN achieves superior overall performance, low false alarm rates, and better minority-class detection.

In future work, we will further investigate adaptive reward design, validation on real-world datasets, and robustness against evolving zero-day attacks.

REFERENCES

[1] Y. E. Tadesse and Y.-J. Choi, "Pattern augmented lightweight convolutional neural network for intrusion detection system," *Electronics*, vol. 13, no. 5, p. Art. no. 932, 2024.

[2] T. Yi, X. Chen, Y. Zhu, W. Ge, and Z. Han, "Review on the application of deep learning in network attack detection," *Journal of Network and Computer Applications*, vol. 212, p. Art. no. 103580, 2023.

[3] K. Sakthi and P. Nirmal Kumar, "A novel attention-based feature learning and optimal deep learning approach for network intrusion detection," *Journal of Intelligent amp; Fuzzy Systems*, vol. 45, no. 3, pp. 5123–5140, 2023.

[4] C. Bunkhumpornpat, K. Sinapiromsaran, and C. Lursinsap, "Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem," in *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 2009, pp. 475–482.

[5] G. E. Batista, R. C. Prati, and M. C. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD explorations newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[6] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artificial Intelligence Review*, vol. 55, no. 1, pp. 453–563, 2022.

[7] G. Caminero, M. Lopez-Martin, and B. Carro, "Adversarial environment reinforcement learning algorithm for intrusion detection," *Computer Networks*, vol. 159, pp. 96–109, 2019.

[8] M. Lopez-Martin, B. Carro, and A. Sanchez-Esguevillas, "Application of deep reinforcement learning to intrusion detection for supervised problems," *Expert Systems with Applications*, vol. 141, p. Art. no. 112963, 2020.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[10] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016, pp. 2094–2100.

[11] Y. Hu, Y. Zhao, Y. Feng, and X. Ma, "Double DQN method for botnet traffic detection system," *Computers, Materials & Continua*, vol. 79, no. 1, pp. 509–530, 2024.

[12] J. S. Jayaprakash, S. Kodati, and A. Kanchana, "An effective cyber security threat detection in smart cities using dueling deep q networks," in *2024 4th International Conference on Mobile Networks and Wireless Communications (ICMNWC)*. IEEE, 2024, pp. 1–5.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[14] T. E. T. Djaidja, B. Brik, S. M. Senouci, A. Boualouache, and Y. Ghamri-Doudane, "Early network intrusion detection enabled by attention mechanisms and rnns," *IEEE Transactions on Information Forensics and Security*, vol. 19, pp. 7783–7793, 2024.

[15] M.-H. Guo, J.-X. Cai, Z.-N. Liu, T.-J. Mu, R. R. Martin, and S.-M. Hu, "Pct: Point cloud transformer," *Computational Visual Media*, vol. 7, no. 2, pp. 187–199, 2021.

[16] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 53–58.

[17] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, pp. 1–12, 2017.

[19] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.

[20] Z. Wang, S. Guo, and G. Liu, "Sa-ddqn: Self-attention mechanism based ddqn for sfc deployment in nfv/mec-enabled networks," in *2022 IEEE 28th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 2023, pp. 720–727.

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.