Mitigating EDoS with Transfer Reinforcement Learning

Amir Javadpour, Forough Ja'fari, Chafika Benzaïd, Tarik Taleb, Fatih Turkmen

***Abstract*—Economic Denial of Sustainability (EDoS) attacks exploit elastic resource provisioning to increase operational expenditure without necessarily causing immediate service unavailability. This paper presents a Q-learning-based mitigation framework for cloud-native and 5G network-slicing environments. The controller observes slice-level telemetry, selects resource-allocation actions, and optimizes a reward that balances quality of service, resource cost, and an EDoS-aware penalty. A transfer Q-learning extension reuses compatible donor Q-tables to reduce the cold-start burden of a newly deployed slice. The current evaluation is a controlled prototype illustration rather than a full 5G slice testbed. The reward curves and scenario tables therefore demonstrate the internal behavior of vanilla and transfer Q-learning under the stated assumptions; literature-derived rows are included only as contextual illustrations and are not treated as head-to-head benchmark results. Within this prototype, transferred knowledge produces earlier reward stabilization and fewer unnecessary allocation actions than training from an empty Q-table. These findings support the feasibility of the proposed control mechanism, while real-world effectiveness, statistical robustness, and cross-platform generalization remain to be established through repeated experiments on an integrated slicing platform. The accompanying demonstration illustrates the resource-allocation and adaptation workflow.**

***Index Terms*—Economic Denial of Sustainability (EDoS), cloud-native security, 5G network slicing, Q-learning, transfer reinforcement learning, adaptive autoscaling**

I. Introduction

Cloud computing provides elastic, pay-as-you-go services, but this flexibility also makes it vulnerable to Economic Denial of Sustainability (EDoS) attacks, where adversaries generate seemingly normal requests that silently inflate costs by exploiting idle capacity and auto-scaling mechanisms. Unlike traditional DoS or Distributed DoS (DDoS) attacks, which disrupt service availability, EDoS undermines financial sustainability and poses long-term risks, particularly for multi-tenant environments. Figure 1 illustrates the attack flow, showing how attackers create fake workloads to trigger scaling and cause unintended or unexpected costs, while also demonstrating the deployment of Artificial Intelligence (AI)-based prevention mechanisms to intercept abnormal patterns and protect both users and resources. With the rise of cloud-native infrastructures and 5G network slicing, which enhance flexibility and scalability, the stealthiness and economic impact of EDoS has become even more critical, threatening the long-term sustainability of modern communication systems.

The effectiveness of traditional defenses, including firewall- and threshold-based controls, remains limited when malicious requests resemble legitimate demand, highlighting the need for adaptive and

Amir Javadpour (Senior Cybersecurity Researcher MOSA!C Lab / ICTFICIAL Oy, Finland). **Forough Ja'fari** (Sharif University of Technology). **Tarik Taleb** (Faculty of Electrical Engineering and Information Technology, Ruhr University Bochum, Bochum). **Chafika Benzaïd** (Faculty of Information Technology and Electrical Engineering, University of Oulu) **Fatih Turkmen** (University of Groningen, The Netherlands)

Corresponding author: Amir Javadpour (a.javadpour87@gmail.com)

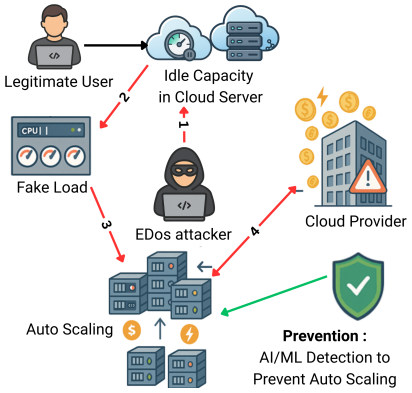

Fig. 1: Economic Denial of Sustainability (EDoS) attack process and prevention.

cost-aware resource control. Early approaches to EDoS detection relied on entropy-based methods [1, 2], which are lightweight but vulnerable to spoofing and manipulation. Supervised ML models like XGBoost [3] have been used to improve robustness in Kubernetes settings, yet have a strong dependency on labeled data and show limited adaptability to novel, unseen attacks. Unsupervised DL approaches, such as VAE and GRU [4], GAN and LSTM [5], and LSTM predictors [6], reduce labeling needs but face instability and concept drift. More recently, FortisEDoS [7, 8] applied deep transfer learning to enhance adaptability, achieving high accuracy but at the cost of heavy computational overhead from its GNN, CNN, and GAT modules.

Motivated by these limitations, this paper introduces an RL-based control framework for mitigating the resource-allocation consequences of EDoS in cloud-native 5G network slicing. The proposed controller is not presented as a standalone attack detector. Instead, it consumes slice telemetry and an externally supplied anomaly indicator and learns whether to preserve, increase, or decrease an allocation. This distinction aligns the contribution with autoscaling mitigation rather than classification. Recent studies have also demonstrated the relevance of RL and federated or multi-agent learning for cloud autoscaling and 5G resource allocation [9, 10, 11, 12]; however, deployment realism, safety constraints, and reproducible evaluation remain open concerns. We provide a demonstration of normal operation, attack-like workload pressure, RL decisions, and transfer-based adaptation. The demonstration illustrates the workflow but is not claimed as an independent experimental benchmark.

Our contributions can be summarized as follows:

- We design a reinforcement learning agent for slice auto-scaling that dynamically responds to traffic conditions while remaining resilient against EDoS-induced misbehavior.
- We propose a transfer Q-learning mechanism to incorporate knowledge from multiple previously trained agents, thereby mitigating the cold-start problem and reducing training overhead.
- We present a controlled prototype comparison between vanilla and transfer Q-learning and use prior EDoS methods only as literature context. We explicitly avoid interpreting cross-paper

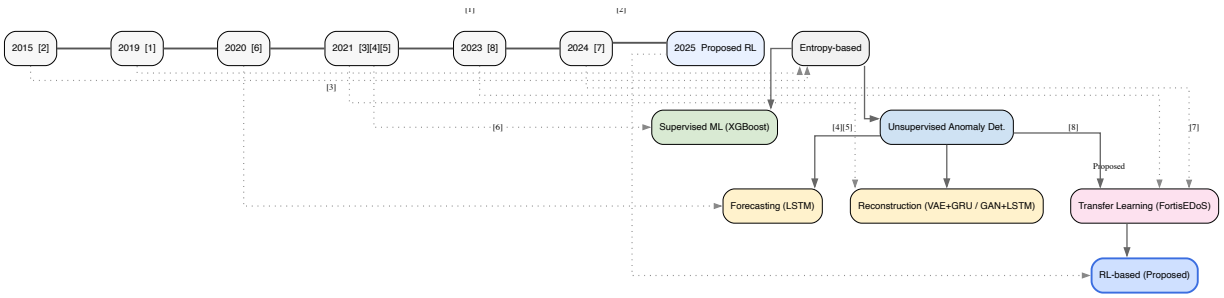

Fig. 2: Timeline and previous works of EDoS detection.

numerical rows as a common experimental benchmark.

The remainder of this paper is structured as follows: Section II reviews advancements in EDoS detection. Section III introduces a reinforcement learning framework for slice-based auto-scaling during EDoS attacks. Section IV discusses the integration of transfer reinforcement learning, while Section V outlines the proposed algorithms. Section VI presents the evaluation process and results, and Section VII concludes this paper.

II. Related work on EDoS

In this section, we review the related work on EDoS detection and mitigation. The timeline of the considered scientific papers is presented in Figure 2.

We first outline early entropy-based methods, then discuss supervised and unsupervised ML approaches, forecasting strategies, and recent advances in deep transfer learning. This review highlights both the progress and the remaining gaps, which motivate our proposed reinforcement learning framework. A summary of the reviewed papers is presented in Table I.

Recent work broadens this context beyond EDoS-specific detection. Zhou et al. survey DRL-based cloud scheduling and identify reproducibility, stability, and realistic workload modeling as persistent challenges [9]. Mishra et al. study RL-based cloud autoscaling as an alternative to fixed Horizontal Pod Autoscaler rules [10]. For 5G slicing, Azimi et al. investigate mobility-aware and energy-efficient federated DRL [11], while Ejaz et al. consider multi-agent DRL for end-to-end slice resource allocation [12]. These studies support the use of adaptive learning for resource control, but they also underscore why results obtained from different platforms cannot be compared numerically without a shared workload, implementation, and protocol.

The study performed by Monge et al. [1] presents one of the earlier attempts to mitigate EDoS attacks in Self-Organizing Networks (SON) by introducing an entropy-based detection method. In this work, the authors exploit entropy metrics to measure the distribution of resource utilization and system responses. Specifically, the framework monitors attributes such as the number of active virtual instances, CPU consumption, and service response time, and any deviation from the expected entropy level is flagged as a possible attack. The approach benefits from its simplicity and low computational overhead, making it suitable for real-time monitoring in virtualized network environments. However, entropy-based techniques are highly dependent on the assumption that normal behavior produces a stable entropy distribution. In dynamic environments, this assumption often fails, and attackers can exploit it by mimicking legitimate traffic patterns, thereby reducing the detection accuracy. This weakness is further highlighted in the work done by Ozçelik and Brooks [2], which demonstrates how entropy-based detectors can be deceived with carefully crafted traffic.

Ozçelik and Brooks [2] analyze the inherent vulnerabilities of entropy-based detection frameworks. Their research shows that attackers can manipulate traffic distribution to match the expected entropy levels of normal operation, effectively bypassing the detection system. The study provides concrete experiments and case studies where spoofing techniques are employed to deceive entropy-based models during DoS attacks. The main contribution of this work lies in its demonstration that entropy is not a reliable standalone metric for anomaly detection, especially when adversaries are adaptive and capable of statistical manipulation. As a result, the study motivates the need for more advanced detection mechanisms that combine entropy with ML or behavioral models. The key limitation is that while it provides a critical evaluation, it does not propose a practical alternative solution, leaving the mitigation challenge open.

TABLE I: Comparison of representative EDoS detection methods.

Ref.	Method / Setting	Advantage and limitation
[1]	Entropy monitoring *Self-organizing networks*	Lightweight online detection; sensitive to spoofing and traffic-distribution shifts.
[2]	Entropy-vulnerability analysis *Manipulated cloud traffic*	Reveals statistical-mimicry attacks; provides no mitigation mechanism.
[3]	XGBoost *Kubernetes/YoYo attacks*	Accurate nonlinear classification; requires labels and may miss unseen attacks.
[4]	VAE–GRU *SDN cloud*	Learns temporal patterns without attack labels; affected by threshold selection and drift.
[5]	GAN–LSTM *SDN cloud*	Captures long-term dependencies; training is unstable and computationally expensive.
[6]	LSTM forecasting *SDN cloud*	Predicts workload evolution; legitimate spikes may generate false alarms.
[7]	Transfer learning with GNN/CNN/GAT *Cloud-native 5G*	Models inter-slice dependencies; incurs substantial computational and scalability costs.
[8]	Extended FortisEDoS *5G-and-beyond slicing*	Adapts across heterogeneous deployments; complex and resource intensive.

The study of Bremler-Barr and David [3] addresses EDoS detection in Kubernetes clusters, which are increasingly deployed in modern cloud-native environments. The authors propose a supervised ML approach based on the XGBoost classifier, trained with labeled datasets that capture normal and attack behaviors. Input features include worker node counts, pod numbers, CPU utilization, and service response times. The experimental results show high detection accuracy, highlighting the effectiveness of gradient boosting methods in modeling complex, nonlinear patterns. This contribution is significant because it addresses autoscaling vulnerabilities in Kubernetes, particularly YoYo attacks where pods are repeatedly scaled up and down, leading to resource exhaustion. However, the limitation lies in its dependence on labeled data, which is costly to collect in real-world deployments. Moreover, supervised models may fail when novel attack strategies emerge that differ from the training data.

The R-EDoS framework presented by Dinh and Park [4] moves towards unsupervised anomaly detection by introducing a hybrid deep learning architecture that combines a Variational Autoencoder (VAE) with a Gated Recurrent Unit (GRU). The VAE is designed to

capture the stochastic distribution of system metrics, while the GRU extracts temporal correlations across time steps in multivariate time-series data. This reconstruction-based approach works by measuring the difference between the original and reconstructed values of key metrics, such as CPU load, memory utilization, and flow statistics in SDN-based clouds. A high reconstruction error indicates anomalous behavior potentially caused by an EDoS attack. The advantage of this model is its ability to learn complex nonlinear dependencies without requiring labeled attack data. Nonetheless, the main challenge lies in setting robust thresholds for anomaly detection and handling concept drift in dynamic traffic environments.

A related unsupervised detection method is introduced by Dinh and Park [5], where the authors leverage Generative Adversarial Networks (GANs) with Long Short-Term Memory (LSTM) models for EDoS detection. In this framework, the GAN generator attempts to reconstruct normal behavior while the discriminator distinguishes between real and generated patterns. The integration of LSTMs enhances the model's ability to capture long-term temporal dependencies. By training the GAN exclusively on normal traffic, any deviation in reconstruction signals the presence of an anomaly. The results demonstrate robustness against common EDoS scenarios in SDN-based clouds. The strength of this approach is its capacity to model highly nonlinear time-series without labeled attack data. However, GAN training is notoriously unstable and computationally expensive, making deployment challenging for real-time scenarios. Furthermore, threshold selection remains a critical issue, as overly strict thresholds can lead to high false positives.

In contrast to reconstruction-based models, the work performed by Dinh and Park [6] adopts a forecasting-based anomaly detection strategy. The authors propose using LSTM networks to predict the future values of network metrics such as CPU usage and bandwidth consumption. The anomaly score is then computed as the deviation between actual observed values and the LSTM forecasts. Large deviations are interpreted as potential EDoS attacks. The forecasting paradigm offers the benefit of explicitly modeling temporal sequences and anticipating future workload changes, which is useful in dynamic environments. Nevertheless, this approach is sensitive to prediction accuracy: if the forecasting model is not well-calibrated, normal workload spikes may be misclassified as attacks. Furthermore, like reconstruction-based methods, forecasting requires a stable notion of "normal" data distribution, which is not guaranteed in fast-changing cloud settings.

FortisEDoS, proposed by Benzaïd et al. [7], represents the most recent advancement in this area, which leverages deep transfer learning to improve EDoS detection in cloud-native network slicing environments. This framework integrates multiple neural network components to capture both temporal and spatial dependencies. Specifically, Graph Neural Networks (GNNs) are used to extract spatial correlations among virtualized network functions, while Convolutional Neural Networks (CNNs) and Graph Attention (GAT) mechanisms handle temporal patterns. By applying transfer learning, FortisEDoS reuses knowledge learned from related tasks, thereby reducing the reliance on large volumes of labeled or normal behavioral data. This is particularly critical in network slicing scenarios, where traffic dynamics vary significantly across slices. The framework is validated through extensive experiments, demonstrating superior detection accuracy and robustness compared to earlier approaches. The limitation is the increased computational overhead due to the combination of multiple deep models, which may hinder real-time applicability. Benzaïd et al. [8] further expand on FortisEDoS by presenting its application in 5G and beyond network slicing environments. The conference version provides detailed implementation insights and benchmarks FortisEDoS against state-of-the-art baselines in real-world-like scenarios. The framework is shown to adapt well to heterogeneous workloads and different cloud-native deployments, benefiting from its deep transfer learning backbone. The results confirm the feasibility of employing advanced spatio-temporal deep learning models for large-scale EDoS detection. However, the reliance on complex deep architectures makes it resource-intensive, raising concerns about scalability when deployed across thousands of slices in large telecom infrastructures.

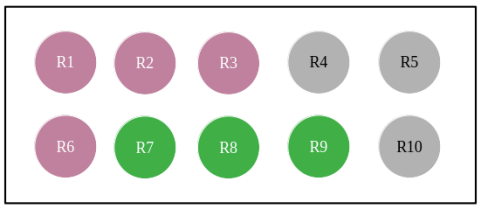

Fig. 3: An example of the action space for an agent in this environment.

III. Reinforcement Learning for Mitigating EDoS

Building on the broader motivation of adaptive EDoS defense, we formulate mitigation as a sequential resource-control problem. The RL component decides allocation changes; it does not itself estimate detection accuracy or false-alarm rate. An anomaly indicator may be supplied by an external detector, policy rule, or monitoring component.

In our proposed method, each deployed slice in 5G utilizes an RL agent that is trained to auto-scale the resources of that slice. To be more precise, the agent decides on which physical resources the slice will be expanded. If the agent selects none of them, it means that the auto-scaling must not be done. If the agent selects more or less resources, it is equivalent to scale-up and scale-down processes, respectively.

Figure 3 is an example of the action space for an agent in this environment. This figure shows a sample network with ten physical resources, R1 to R10. Two slices are currently deployed on this network and their dedicated resources are shown in purple and green. The slice specified with green is a newly deployed one.

In the example shown in Figure 3, the action space of the agent is a binary number with ten bits. The ith bit is zero in agent's action, if it decides to not dedicate Ri for that slice. For example, if the agent wants to show that no scaling must be done, the action is **111**00**1**0000. As another example, if the agent decides to scale-up the resources by including R5 in the dedicated resource of the purple slice, the action is **111**0**11**0000.

IV. Transfer Reinforcement Learning for Mitigating EDoS

For the transfer-learning mechanism illustrated in Figure 4, Q-learning is used as the tabular control algorithm. Previously trained agents may share Q-tables with a newly deployed slice to reduce cold-start exploration. Transfer is permitted only when donor and target slices use compatible state discretization, action indexing, reward semantics, and safety constraints. A common infrastructure alone does not guarantee compatibility; incompatible entries must be excluded or mapped before aggregation.

The formula for updating the Q values for a single agent is as follows, where $Q(s, a)$ is the Q value of performing action a in state s, s' is the state that the agent will be in after performing action a in state s, $Q'(s')$ is the maximum Q value of state s', α is the learning rate, γ is the discount factor, and r is the reward value.

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a')) \tag{1}$$

Now, to include the knowledge of other agents in this formula, we propose to consider a factor, called transfer factor (β), and update the above equation as the below one, where Q' is the Q value of the agent trained in the old slice.

$$\begin{aligned} Q(s,a) = (1-\beta)[(1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))] \\ + \beta Q'(s,a) \end{aligned} \tag{2}$$

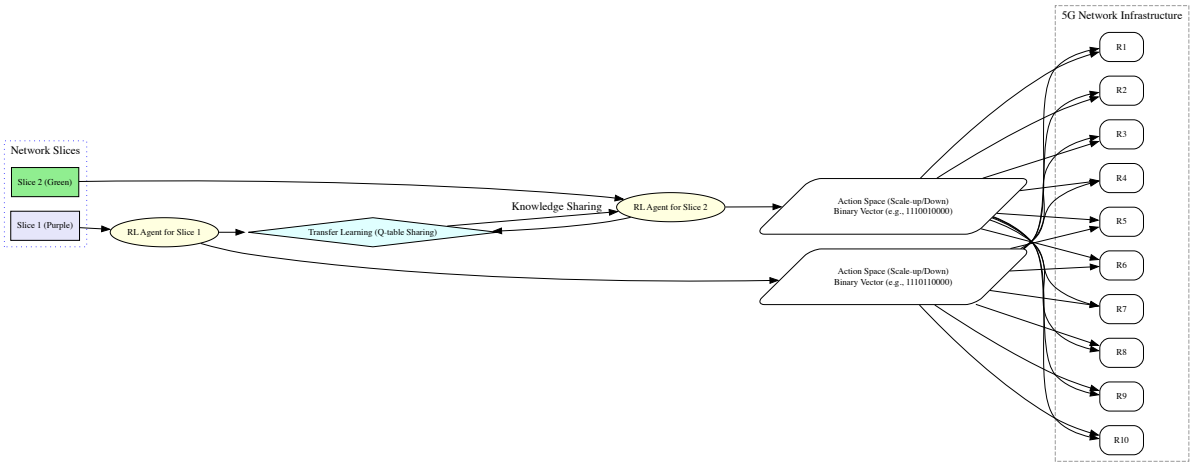

Fig. 4: Reinforcement Learning and Transfer Learning-based framework for mitigating EDoS.

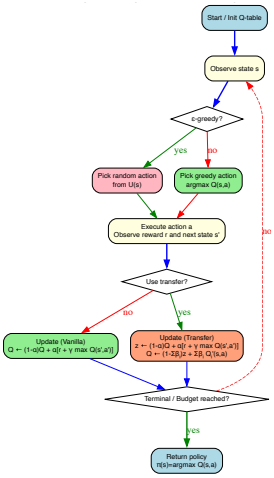

Fig. 5: Flowchart of the RL agent training process with transfer learning option.

The transfer factor satisfies $0 < \beta < 1$; larger values place greater weight on donor knowledge and therefore require stronger compatibility checks.

In order to include the knowledge of more than one agent, we will assume the following equation.

$$Q(s,a) = (1 - \sum \beta_i)[(1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a'))] + \sum[\beta_i Q'_i(s,a)] \tag{3}$$

Here, β_i is the transfer weight of donor i and Q'_i is its Q value. The constraint $0 < \sum_i \beta_i < 1$ preserves a nonzero contribution from the target agent's own update.

The training process of an agent in our proposed framework is presented in Figure 5.

A. State Space Definition

The state space represents the observable parameters by the RL agent. In our framework, each state $s \in S$ is characterized by a tuple of network and slice-level features such as incoming traffic rate T, CPU utilization U_{CPU}, memory utilization U_{Mem}, latency L, and quality of service (QoS) score Q_{os}. Formally, the state can be expressed as:

$$s = (T, U_{\text{CPU}}, U_{\text{Mem}}, L, Q_{\text{os}}).$$

These parameters describe the operating condition available to the controller. They do not, by themselves, establish that traffic is malicious. When an anomaly score is used, it is treated as an external input produced by a monitoring or detection component. Therefore, the present method is evaluated as an autoscaling mitigation policy rather than as an EDoS classifier.

B. Action Space Representation

The action space A is represented as a binary vector, where each bit corresponds to a physical resource R_i. If $a \in A$ denotes the action taken by the agent, then:

$$a = (b_1, b_2, \ldots, b_n), \quad b_i \in \{0, 1\},$$

where n is the total number of physical resources. A value of $b_i = 1$ indicates that resource R_i is allocated to the slice, whereas $b_i = 0$ denotes no allocation. Scale-up actions correspond to switching b_i from 0 to 1, scale-down actions correspond to switching from 1 to 0, and a no-action decision implies that the binary vector remains unchanged.

C. Reward Function Design

The reward function $r(s,a)$ is designed to balance service quality, resource efficiency, and resilience against EDoS. It can be mathematically formulated as:

$$r(s,a) = \lambda_1 \cdot Q_{\text{os}} - \lambda_2 \cdot C(a) - \lambda_3 \cdot P_{\text{EDoS}}(s,a),$$

where Q_{os} represents the achieved QoS level, $C(a)$ is the cost of allocating resources determined by action a, and $P_{\text{EDoS}}(s,a)$ denotes the penalty incurred if the scaling decision is influenced by malicious EDoS traffic. The coefficients $\lambda_1, \lambda_2, \lambda_3 > 0$ are weight factors that tune the trade-off between efficiency and robustness.

D. Action–State–Reward Mapping

To show the dynamics of the proposed RL framework, the mapping between states, actions, and rewards can be described as follows:

- If T is high and $P_{\text{EDoS}}(s,a) = 0$, then a scale-up action increases Q_{os} and yields a positive reward.
- If T is low and the agent performs scale-down, $C(a)$ decreases and the reward remains positive.
- If $P_{\text{EDoS}}(s,a) > 0$ but the agent does not scale-up, then the penalty is avoided and a positive reward is received.
- If the agent reacts to EDoS by scaling-up unnecessarily, the term $\lambda_3 \cdot P_{\text{EDoS}}(s,a)$ dominates and the reward becomes negative.

Algorithm 1 EDoS-Aware State, Action, and Reward Processing

Require: Telemetry $(T_t, U_t^{cpu}, U_t^{mem}, L_t, Q_t^{os}, z_t)$, allocation x_t, executed action a_t, weights $\lambda_1, \lambda_2, \lambda_3$
Ensure: State s_t, admissible actions $\mathcal{U}(s_t)$, and reward r_t when the next observation is available
Normalize telemetry and compute $A_t \leftarrow f_{\text{anom}}(z_t)$
$s_t \leftarrow (\hat{T}_t, \hat{U}_t^{cpu}, \hat{U}_t^{mem}, \hat{L}_t, \hat{Q}_t^{os}, A_t)$
$\mathcal{U}(s_t) \leftarrow \{\text{no-change}\}$
for $i = 1$ **to** n **do**
  Add the feasible single-bit flip of $x_t[i]$ to $\mathcal{U}(s_t)$
if the next observation is available **then**
  $C_t \leftarrow C(a_t)$; $P_t \leftarrow g(\text{scale\_up}(a_t), A_{t+1})$
  $r_t \leftarrow \lambda_1 Q_{t+1}^{os} - \lambda_2 C_t - \lambda_3 P_t$
return $s_t, \mathcal{U}(s_t), r_t$

E. Scenario Illustration

Consider a network with two slices: one serving legitimate traffic and another under an EDoS attack. In the legitimate slice, the state $s = (T, U_{\text{CPU}}, U_{\text{Mem}}, L, Q_{\text{os}})$ indicates high load and low anomaly, prompting the agent to choose a scale-up action with reward:

$$r = \lambda_1 Q_{\text{os}} - \lambda_2 C(a).$$

In the attacked slice, the same high traffic T is accompanied by an anomaly score indicating EDoS. The agent learns to keep the action unchanged, avoiding unnecessary costs. In this case, the reward is:

$$r = \lambda_1 Q_{\text{os}} - \lambda_3 P_{\text{EDoS}}(s, a),$$

which ensures sustainability and economic efficiency.

F. Complexity Analysis

The nominal binary allocation space contains 2^n vectors; enumerating it would be impractical as n grows. The proposed action-masking procedure therefore restricts each decision to no change or a single feasible bit flip, yielding at most $n + 1$ candidate actions and $O(n)$ action evaluation per update. Aggregating k compatible donor values adds $O(k)$ work for the selected state–action pair, giving $O(n + k)$ update time under the masked representation. The tabular memory requirement remains $O(|S||A_{\text{masked}}|)$ and can still become large when continuous telemetry is finely discretized. Consequently, scalability beyond modest state spaces would require function approximation or state aggregation.

V. Proposed Algorithms

The seven operational procedures of the framework are consolidated into three algorithms to avoid repetition while preserving the complete workflow. Algorithm 1 constructs the EDoS-aware state, generates safe local actions, and computes the reward. Algorithm 2 performs either vanilla or transfer Q-learning through one unified training routine. Algorithm 3 applies the learned policy under safety constraints and records the metrics required for deployment analysis and future reproducible evaluation.

Algorithm 1 merges state extraction, action masking, and reward computation. The anomaly value remains an externally supplied monitoring signal; therefore, this procedure supports mitigation decisions rather than performing EDoS classification.

Algorithm 2 treats vanilla Q-learning as the zero-transfer case and activates donor aggregation only when state discretization, action indexing, reward semantics, and safety constraints are compatible. This formulation removes duplicate training loops and makes the methodological difference explicit.

Algorithm 2 Unified Vanilla and Transfer Q-Learning

Require: Mode $m \in \{\text{vanilla}, \text{transfer}\}$, compatible donor tables $\{Q_i'\}$, weights $\{\beta_i\}$, $\alpha, \gamma, \varepsilon$, episodes E, horizon T
Initialize $Q(s, a) \leftarrow 0$
for $e = 1$ **to** E **do**
  Reset the environment and obtain s and $\mathcal{U}(s)$ from Algorithm 1
  for $t = 1$ **to** T **do**
    Select a by ε-greedy search over $\mathcal{U}(s)$
    Execute a; observe s' and compute r using Algorithm 1
    $z \leftarrow (1 - \alpha)Q(s, a) + \alpha\big(r + \gamma \max_{a' \in \mathcal{U}(s')} Q(s', a')\big)$
    if $m = \text{transfer}$ and donor compatibility is satisfied **then**
      $Q(s, a) \leftarrow (1 - \sum_i \beta_i) z + \sum_i \beta_i Q_i'(s, a)$
    else
      $Q(s, a) \leftarrow z$
    $s \leftarrow s'$
return $\pi(s) = \arg\max_{a \in \mathcal{U}(s)} Q(s, a)$

Algorithm 3 Safe Policy Execution and Evaluation

Require: Policies Π, safety guard $\mathcal{S}$, scenarios $\mathcal{L}$, evaluation windows E
for all $\pi \in \Pi$ **do**
  for all $\ell \in \mathcal{L}$ **do**
    for $e = 1$ **to** E **do**
      Reset scenario ℓ
      while the service or evaluation window is active **do**
        Obtain s and $\mathcal{U}(s)$ from Algorithm 1
        $a^\star \leftarrow \arg\max_{a \in \mathcal{U}(s)} Q_\pi(s, a)$
        if $\mathcal{S}$ blocks $a^\star$ **then** replace it with a safe action
        Apply $a^\star$ and log QoS, cost, latency, waste, resilience, and energy proxy
      Compute ESI and store the window-level results
return Deployment logs and evaluation summaries

Algorithm 3 combines the former deployment and evaluation procedures. In the current prototype, it supports only the internally comparable vanilla and transfer Q-learning cases; applying it as a common benchmark requires fixed configurations, repeated seeded runs, and dispersion statistics.

VI. Experimental Evaluation

This section reports a controlled prototype illustration of the proposed decision mechanism. It does not constitute a full network-slice deployment or a statistically validated comparison with external methods. The only internally comparable conditions are vanilla Q-learning and transfer Q-learning, which use the same conceptual state, action, and reward definitions.

A. Prototype scope and reproducibility boundaries

The prototype is represented through illustrative workload and telemetry sequences covering legitimate demand changes and EDoS-

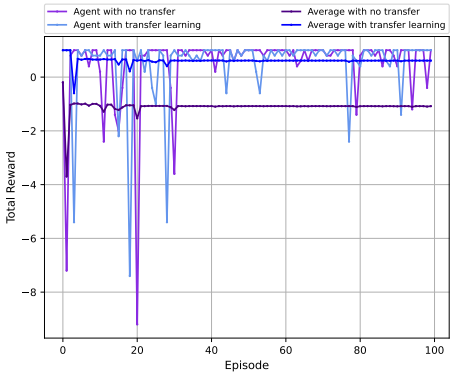

Fig. 6: Comparison of RL agents with and without transfer learning.

like pressure on an elastic service. No public packet-level or flow-level dataset is claimed, and the current implementation is not integrated with a 5G core, Kubernetes cluster, or network-slice orchestrator. The available prototype record does not contain a complete run manifest recording fixed numerical values for α, γ, the exploration schedule ε, reward weights λ_1–λ_3, donor weights β_i, random seeds, or repeated-run outputs. We therefore do not retrospectively assign parameter values or claim reproducibility that cannot be supported. The mathematical constraints are $0 < \alpha \leqslant 1$, $0 \leqslant \gamma < 1$, $0 \leqslant \varepsilon \leqslant 1$, $\lambda_i > 0$, and $0 < \sum_i \beta_i < 1$. A reproducible empirical extension must publish the exact configuration, source code, workload generator, platform versions, seeds, train/evaluation split, number of repetitions, and dispersion statistics.

B. Interpretation of reported quantities

The numerical entries in Tables II and III are presented as illustrative scenario values. They are not measurements produced by re-running the cited methods. "Reward" is the scalar return defined by the reward function; "latency" is the scenario response-time indicator; "cost" is a relative resource-consumption category; "energy index" is a unitless proxy rather than a physical joule measurement; "waste" is the percentage of allocated resource-time not used by legitimate service demand; "QoS" is the percentage of decision intervals satisfying the scenario service target; and "resilience" is the percentage of attack-affected intervals in which unnecessary scale-up is avoided while the QoS target is maintained. The ESI values are a normalized illustrative economic-sustainability score used in the prototype display; because the prototype record does not include the normalization constants, ESI is not used for inferential or cross-paper claims.

C. Comparison of total reward

Figure 6 shows the prototype reward trajectories over 100 displayed episodes. Under this single illustrative configuration, the transfer-initialized agent reaches a stable high-reward region earlier than the agent initialized without donor knowledge. Because repeated runs, random seeds, and variance estimates are unavailable, the figure should be interpreted as evidence of the intended cold-start behavior in this prototype, not as a statistically significant performance estimate.

D. Comparison with baseline methods

Table II presents numerical rows for several method categories, but they do not originate from a common experimental environment. The rows associated with Refs. [1, 2, 3, 4, 5, 6, 7, 8] were not executed under the present prototype and must not be read as apples-to-apples baseline results. They provide only a qualitative display of method categories. The internally meaningful observation is limited to the displayed vanilla and transfer Q-learning scenarios: transfer initialization is associated with earlier stabilization and lower illustrative waste in the same prototype representation.

E. Analysis under attack intensity

Table III presents illustrative high-intensity scenario values. The cited-method rows were not regenerated in the present environment and therefore cannot establish relative robustness. Within the two internally represented Q-learning cases, the transfer-based policy displays a higher illustrative reward and fewer unnecessary allocations than the empty-table initialization. This observation is hypothesis-generating and requires repeated evaluation in an integrated environment.

F. Comparative perspective with state-of-the-art

Table I summarizes distinct EDoS detection and mitigation paradigms. The present study does not empirically establish faster convergence or higher resilience than FortisEDoS or any other external method because the implementations, datasets, platforms, and metrics differ. Its contribution is instead the formulation of EDoS-aware autoscaling as a transfer Q-learning control problem and the demonstration of its intended behavior in a limited prototype. A common benchmark would be required before making quantitative state-of-the-art claims.

G. Comparative Discussion

Conceptually, the proposed framework replaces fixed scaling rules with a policy that optimizes a multi-objective reward. Transfer Q-learning can reduce early exploration when donor and target slices are compatible. The current evidence supports this mechanism only at prototype level; it does not demonstrate production-grade service assurance, attack detection, or generalization to unseen infrastructures. Safety guards, donor-selection criteria, and negative-transfer detection must be validated before deployment.

Figure 7 illustrates the extended resource allocation and scaling process enhanced with reinforcement learning (RL).

Initially, two services run under normal conditions with balanced resource distribution among legitimate users (A). As demand grows, the system performs a scale-up action to accommodate additional load (B). When a traffic flood occurs due to malicious activity (C), both users and attackers consume resources simultaneously. After the attack subsides, idle resources remain in the system, reducing efficiency (D). At this stage, the RL agent is activated to decide intelligent scale-up and scale-down actions (E). The agent receives feedback through rewards, such as minimizing idle time, which guides its learning process (F). Finally, in cases where insufficient traffic is available for analysis, the agent acknowledges the lack of information and adapts accordingly (G).

VII. Conclusion

This paper formulated EDoS mitigation as a resource-control problem and proposed vanilla and transfer Q-learning policies for slice autoscaling. The controller combines service telemetry, resource cost, and an externally supplied EDoS-related signal in its reward, while compatible donor Q-tables provide a mechanism for reducing cold-start exploration. The framework is evaluated as mitigation rather than detection: no detection accuracy or false-alarm rate is measured. The evaluation is a controlled prototype illustration. The reward curve and scenario tables suggest that transfer initialization can stabilize the policy earlier and reduce unnecessary allocations under the displayed assumptions, but the absence of an integrated slice environment, complete hyperparameter manifest, repeated runs, and variance estimates prevents statistical or real-world performance claims. Literature rows are used only as contextual illustrations rather than direct baselines. Future work should implement the controller in

TABLE II: Illustrative prototype values and literature-context rows. Cited methods were not re-run under a common setup; the table is not a head-to-head benchmark.

Method	Conv. Episodes	Avg. Reward	Latency (ms)	Cost	Energy index	Waste (%)	QoS (%)	Resilience (%)	ESI
Entropy-based [1]	–	2.0	85	Low	190	40	65	35	0.50
Entropy vulnerability [2]	–	2.2	82	Low	185	38	66	32	0.48
Supervised ML (XGBoost) [3]	–	3.2	65	High	195	35	75	55	0.70
VAE and GRU [4]	700	4.8	60	Medium	175	25	80	68	1.05
GAN and LSTM [5]	650	5.0	58	Medium	172	23	82	70	1.10
Forecasting LSTM [6]	620	4.9	57	Medium	170	22	81	69	1.08
FortisEDoS [7]	600	5.1	55	Medium	170	22	82	72	1.15
FortisEDoS (improved) [8]	580	5.3	54	Medium	168	21	83	73	1.18
Vanilla Q-learning (this work)	500	5.8	52	Medium	165	20	85	75	1.20
Transfer Q-learning (this work)	200	7.1	47	Low	150	8	92	90	1.80

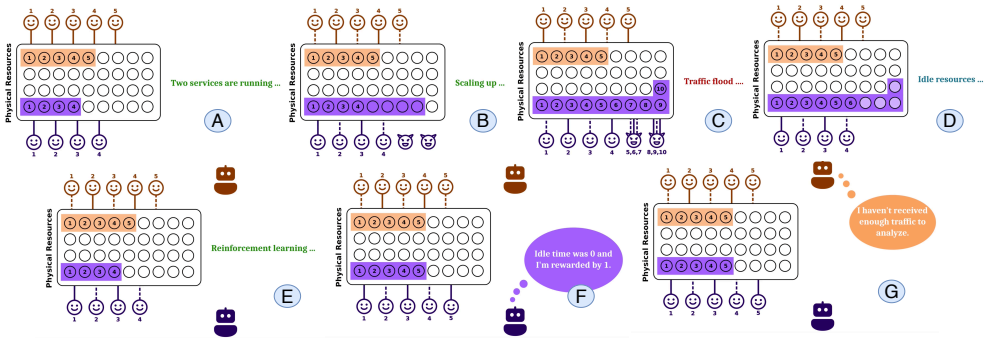

Fig. 7: Resource allocation and scaling process with reinforcement learning: (A) normal service execution; (B) scale-up action under increased demand; (C) traffic flood caused by attackers; (D) idle resources after attack; (E) reinforcement learning agent takes control; (F) agent receives reward for minimizing idle time; (G) insufficient traffic observed for analysis (**EDoS Demo**)

TABLE III: Illustrative high-intensity scenario values; external methods were not re-implemented in a common environment.

Method	Attack Level	Reward	Waste (%)	QoS (%)	Resilience (%)
Entropy-based [1]	High	1.9	60	55	25
Supervised ML (XGBoost) [3]	High	3.0	40	70	50
GAN and LSTM [5]	High	4.4	30	75	65
FortisEDoS [7]	High	4.4	30	75	65
Vanilla Q-learning (this work)	High	4.7	30	78	60
Transfer Q-learning (this work)	High	6.5	12	88	85

a reproducible Kubernetes or 5G slicing testbed, publish the complete workload generator and configuration, evaluate repeated seeded runs, study negative transfer and safety constraints, and report cost, QoS, latency, and energy using physically grounded measurements.

Acknowledgment

The work in this paper was supported in part by the Federal Ministry of Research, Technology, and Space (BMFTR), Germany, through the Project 6GEM+ under Grant 16KIS2411; and in part by the European Union through the 6G-Path project under Grant 101139172.

References

[1] M. A. S. Monge, J. M. Vidal, and G. M. Pérez, "Detection of economic denial of sustainability (EDoS) threats in self-organizing networks," *Computer Communications*, vol. 145, pp. 284–308, 2019.

[2] I. Ozçelik and R. R. Brooks, "Deceiving entropy-based dos detection," *Computers & Security*, vol. 48, pp. 234–245, Feb. 2015.

[3] A. Bremler-Barr and R. B. David, "Kubernetes autoscaling: YoYo attack vulnerability and mitigation," in *Proceedings of the 11th International Conference on Cloud Computing and Services Science (CLOSER)*, 2021, pp. 34–44.

[4] P. T. Dinh and M. Park, "R-edos: Robust economic denial of sustainability detection in an sdn-based cloud through stochastic recurrent neural network," *IEEE Access*, vol. 9, pp. 35 057–35 074, Feb. 2021.

[5] ——, "Economic denial of sustainability (edos) detection using gans in sdn-based cloud," in *Proc. of the 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE)*, Jan. 2021, pp. 135–140.

[6] ——, "Dynamic economical-denial-of-sustainability (edos) detection in sdn-based cloud," in *Proc. of the Fifth International Conference on Fog and Mobile Edge Computing (FMEC)*, Apr. 2020, pp. 62–69.

[7] C. Benzaïd, T. Taleb, A. Sami, and O. Hireche, "Fortisedos: A deep transfer learning-empowered economical denial of sustainability detection framework for cloud-native network slicing," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 4, pp. 2818–2835, Aug. 2024.

[8] ——, "A deep transfer learning-powered edos detection mechanism for 5g and beyond network slicing," in *Proc. of IEEE GLOBECOM*, Kuala Lumpur, Malaysia, Dec. 2023, pp. 4747–4753.

[9] G. Zhou, W. Tian, R. Buyya, R. Xue, and L. Song, "Deep reinforcement learning-based methods for resource scheduling in cloud computing: A review and future directions," *Artificial Intelligence Review*, vol. 57, no. 5, p. 124, 2024.

[10] P. Mishra, S. Hans, D. Saha, and P. Moogi, "Optimizing cloud workloads: Autoscaling with reinforcement learning," in *2024 IEEE International Conference on Cloud Computing (CLOUD)*, 2024, pp. 217–222.

[11] Y. Azimi, S. Yousefi, H. Kalbkhani, and T. Kunz, "Mobility aware and energy-efficient federated deep reinforcement learning assisted resource allocation for 5g-ran slicing," *Computer Communications*, 2024.

[12] M. A. Ejaz, G. Wu, and T. Iqbal, "Dynamic and efficient resource allocation for 5g end-to-end network slicing: A multi-agent deep reinforcement learning approach," *International Journal of Communication Systems*, vol. 37, no. 17, p. e5916, 2024.