GORGO: A Reinforcement Learning Model for DoS Vulnerability Analysis on 5G and Beyond Networks

Konstantinos Giapantzis[1*], Apostolis Zarras[1], Sarantis Kalafatidis[2], Antonios Lalas[2], Konstantinos Votis[2], Christos Xenakis[1]

[1*]Department of Digital Systems, University of Piraeus, Karaoli & Dimitriou 80, Piraeus, 18534, Greece.
[2]Information Technologies Institute, Centre for Research & Technology, Hellas, 6th km Charilaou-Thermi Rd, Thessaloniki, 57001, Greece.

*Corresponding author(s). E-mail(s): kostasgiap@unipi.gr;

Abstract

The present research introduces GORGO, a unique penetration testing tool that utilizes Reinforcement Learning. It performs vulnerability assessments on 5G—and beyond—infrastructures against sophisticated Denial of Service (DoS) attacks driven by Artificial Intelligence (AI). GORGO consists of multiple Deep Q-Learning agents working in collaboration to assess the resilience of target infrastructure, making it the only penetration testing tool capable of performing such comprehensive evaluations. The objective is to construct the optimal attack strategy and provide the information to understand and mitigate AI-driven DoS attacks. The rationale is to provide detailed insights into the extent of Quality of Service degradation and to identify which strategic choices had the most significant impact on the target infrastructure. GORGO was evaluated in one real Software Defined Network (SDN)-based Cloud Native 5G infrastructure and one Beyond 5G (B5G) testbed. The experimental results demonstrate that GORGO is a highly efficient Reinforcement Learning model and successfully uncovered vulnerabilities in both testing environments.

Keywords: Penetration Testing, 5G, Vulnerability Assessment, Reinforcement Learning, AI-driven DoS

1 Introduction

The 5G Core (5GC), as outlined by the 3rd Generation Partnership Project (3GPP) [1], marks a substantial step forward in network design, bringing significant improvements over previous generations. The key features of 5G technology include faster data transmission, reduced latency, increased capacity, and enhanced connectivity, all of which contribute to seamless communication within the 5G infrastructure while ensuring compliance with regulatory requirements [2]. However, despite these advances, 5G networks remain exposed to risks, particularly when internal services become targets for malicious actors [3]. The risk is further aggravated by the growing use of AI, particularly adversarial neural networks that are capable of launching DoS attacks targeting 5G and beyond network technologies (called from now on 5G+) [4]. AI-driven DoS attacks can significantly disrupt 5G services by exploiting existing vulnerabilities in network infrastructures [5]. The capacity of AI systems to execute complex and coordinated attacks across multiple protocols highlights the vulnerability of modern infrastructures to AI-based threats [6].

Various approaches have been suggested to address these security issues and enhance the resilience and reliability of 5G networks against emerging threats [7]. Advanced AI and machine learning methods can detect and counteract attacks in real time by using pattern recognition and predictive analytics to identify unusual network traffic behavior [8] Robust security protocols, including improved authentication techniques and secure communication frameworks, safeguard critical network components from unauthorized interference. Enhancements in network architecture and protocol design, such as network slicing [9] and virtualization technologies [10], help isolate different segments of the network, limiting the spread of attacks and ensuring uninterrupted services.

Despite these advancements, current solutions struggle against sophisticated attack strategies, particularly those leveraging AI capabilities [11]. Attackers using AI-based approaches like Evil-GPT [12] and WOLFGPT [13] can adapt their tactics to bypass detection mechanisms [11]. Traditional security tools [14–16], commonly used for evaluating system resilience, often rely on predefined rules and standard detection methods, making them ineffective against adaptive and intelligent threats. Existing solutions [17, 18] also lack in-depth insights into the specific impact of DoS attacks on the Quality of Service (QoS) of 5G components. Understanding how DoS attacks affect network resilience and reliability is crucial for uncovering vulnerabilities that conventional tools might miss. Yet, developing mitigation measures is difficult without thoroughly examining how these attacks degrade service performance. Therefore, there is an urgent need for security solutions that match the sophistication of AI-powered attacks, underscoring the importance of creating adaptive and intelligent defense mechanisms to secure 5G infrastructure against evolving threats.

To address these challenges, we introduce GORGO, a multi-agent Reinforcement Learning algorithm designed to uncover vulnerabilities in real SDN-based 5G+ infrastructures. GORGO employs intelligent attack strategies to identify and exploit potential network weaknesses. Utilizing Deep Q-Network (DQN) agents and extensive experiments, we demonstrate its effectiveness in evaluating network resilience against

advanced threats, particularly AI-based DoS attacks. GORGO stands apart from conventional penetration testing solutions by modeling complex DoS attack scenarios and providing deeper insights into network weaknesses. By integrating DoS attacks with protocol-level fuzzing techniques, GORGO generates custom network packets tailored to specific 5G services. This method uncovers vulnerabilities other tools may overlook, offering a comprehensive assessment of the network's capabilities and the communication protocols its services utilize. GORGO is crucial in enhancing Intrusion Detection Systems (IDS) by generating data based on sophisticated attack strategies. Analyzing these malicious packets enables AI-based IDS to train more effectively, improving real-time detection and response capabilities. As an evaluation tool, GORGO provides comprehensive reports detailing the attack structure, strategy, and impact metrics such as latency, throughput, and packet loss. This allows network administrators to accurately assess the system's QoS before and after attacks, supporting measures to enhance system resilience and defensive capabilities.

In summary, we make the following main contributions:

- We present GORGO, the first application of DQN agents capable of autonomously learning and collaborating to evaluate the resilience of real 5G+ networks against DoS attacks, rather than relying on simulated environments. GORGO adjusts its strategies based on the application context and the targeted QoS, utilizing distinct attack tactics for each 5G service to efficiently uncover vulnerabilities.
- We extend GORGO's capabilities beyond DoS evaluation by allowing it to autonomously generate custom packets for protocol-level fuzzing, enhancing the sophistication and efficacy of vulnerability detection.
- We propose GORGO as a penetration testing solution, providing system administrators with the ability to assess their networks' resilience against advanced AI-driven attacks.

The remainder of this article is organized as follows. Section 2 provides background information on Reinforcement Learning, particularly Q-learning and Deep Q-learning, as well as an overview of the 5G core network architecture, focusing on key elements. Section 3 presents the network exposure considerations adopted in this research, defining the context under which GORGO operates and potential attack scenarios. Section 4 details GORGO's Reinforcement Learning methodology, including its interaction with the 5G+ environment, reward calculation, autonomous agent roles and neural network architecture. Section 5 provides the evaluation methodology, describes the experimental setups used to test GORGO and analyzes results obtained from various real-world 5G and B5G testbeds. Section 6 presents a comparative analysis of GORGO against existing penetration testing tools, highlighting its unique strengths and improvements. Section 7 discusses the insights gained from multi-agent Deep Q-Learning approach, including attack precision, payload content analysis and cross-component transfer learning, while it outlines the limitations of the current evaluation and proposes future work to address those gaps. Section 8 provides an overview of related research on attacks targeting 5G and B5G infrastructures. Finally, Section 9 concludes the article, summarizing key findings and highlighting potential directions for future research.

2 Background

This section presents an overview of Reinforcement Learning and its foundational algorithms. It begins with introducing the fundamental concepts and mathematical formulation of Reinforcement Learning, modeled as Markov Decision Process (MDP). It then discusses Q-learning, a model-free Reinforcement Learning algorithm that enables agents to learn optimal policies through iterative updates based on experience. Finally, it introduces Deep Q-learning, which extends traditional Q-learning by incorporating deep neural networks to approximate the action-value function, allowing agents to handle complex environments with large or continuous state spaces.

Regarding the 5G, an overview of the core network architecture is presented, explaining how it uses network slicing and virtualization technologies to create flexible and efficient networks. It highlights the key components like the Session Management Function (SMF), Access and Mobility Management Function (AMF) and User Plane Function (UPF), which handle tasks such as session management, mobility, and data forwarding.

2.1 Reinforcement Learning

Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment [19]. Unlike supervised learning [19], which relies on labeled data, Reinforcement Learning involves learning optimal actions through trial and error to maximize cumulative rewards. Developed to address sequential decision-making tasks [20], the core idea is to deploy a policy that maps inputs to actions within an environment. The policy is improved over time by exploring different actions and observing associated feedback signals, known as rewards. This approach allows the agent to adapt its strategy to maximize cumulative rewards, making Reinforcement Learning particularly effective. This machine learning approach can be applied to a variety of tasks, such as dynamic traffic routing and optimization [21], network slicing within 5G infrastructures [22], and the integration with Digital Twin technology [23].

To validate these concepts, the mathematical formulation of Reinforcement Learning is presented, which encapsulates how an agent learns optimal actions by interacting with an environment. Reinforcement Learning tasks are often modeled as a MDP [24], defined by a tuple $(S_t, A_t, \pi_t, R_{t+1}, \gamma)$, where at each time step t:

- S_t is a representation of the environment's state, $S_t \in \mathcal{S}$, and $\mathcal{S}$ is the set of possible states.
- A_t is an action, $A_t \in \mathcal{A}(S_t)$, and $\mathcal{A}(S_t)$ is the set of actions available in state S_t.
- π_t is the policy and $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$.
- R_{t+1} is the numerical reward received by the agent as a consequence of its action, and $R_{t+1} \in \mathcal{R} \subset \mathbb{R}$.
- $\gamma \in [0, 1]$ is the discount rate representing the present value of the future rewards.

A particular finite MDP is defined by its state and action sets and by the one-step dynamics of the environment. Given any state s and action a, the probability of each possible pair of next state s' and reward r is denoted [25]:

$$p(s', r|s, a) = Pr\{S_{t+1} = s', R_{t+1} = r|S_t = s, A_t = a\} \tag{1}$$

The additional required concept is discounting. According to MDP, the agent tries to select actions to maximize the sum of the discounted rewards it receives over the future. In particular, it chooses A_t to maximize the expected discounted return [26]:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{2}$$

In Reinforcement Learning algorithms, value functions estimate how beneficial it is for an agent to be in a particular state and are defined with respect to specific policies. ForMDPs the state-value function $\upsilon_p(s)$ for policy π can be defined as [27]:

$$\upsilon_p(s) = \mathbb{E}_\pi[G_t|S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s] \tag{3}$$

where:

- $\mathbb{E}_\pi[.]$ denotes the expected value of a random variable given that the agent follows policy π and t is any time step.
- G_t is the discounted return.

Similarly, the value of taking action a in state s under a policy π, denoted $q_\pi(s, a)$, is defined as the expected return starting from s, taking the action a, and thereafter following policy π [25]:

$$\begin{aligned} q_\pi(s, a) &= \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right] \\ &= \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \middle| S_t = s, A_t = a \right] \end{aligned} \tag{4}$$

where, q_π is the the action-value function for policy π. The value functions υ_π and q_π can be estimated from experience. For instance, if an agent follows the policy π and keeps an average of the actual returns observed after each state is visited, this average will converge to the state value, $\upsilon_\pi(s)$, as the number of visits to that state approaches infinity. Likewise, if separate averages are maintained for each action taken in a state, these averages will converge to the action values, $q_\pi(s, a)$ [25].

Solving a Reinforcement Learning task involves finding a policy that maximizes long-term rewards. Specifically, a policy π is considered better than or equal to another policy π' if its expected return is greater than or equal to that of π' for every state $s \in S$, meaning $\pi \geq \pi'$ only if $\upsilon_\pi(s) \geq \upsilon_{\pi'}(s)$ for all s. There always exists at least one optimal policy, denoted by π^*, that is better than or equal to all other policies. To summarize, in finite MDPs, value functions precisely define an optimal policy by creating a partial ordering of all feasible policies. Although there may be multiple optimal policies, they all share the same optimal state-value function υ_* defined as [28]:

$$v_*(s) = \max_{\pi} v_\pi(s), \ \forall \ s \in S \tag{5}$$

In addition, Reinforcement Learning algorithms are commonly grouped according to the characteristics of their action space [29]. Continuous action environments typically require policy gradient methods, such as Proximal Policy Optimization (PPO) [30], because of their effectiveness in handling high-dimensional and continuous action distributions. In contrast, environments with a finite and discrete set of actions, such as our protocol selection, thread count, request rate, inter-request timing, session management, packet size and fuzzing parameters, are well suited to value-based methods like DQN [31]. DQN combines Q-learning with deep neural networks to approximate value functions efficiently in large discrete action spaces, providing strong stability and performance [32]. Accordingly, we implement each agent in GORGO using DQN to take advantage of these benefits in discrete action Reinforcement Learning settings. In the subsections that follow, we examine the theoretical foundations and practical implementations of both Q-learning and Deep Q-Learning.

2.1.1 Q-learning

Building upon the foundational concepts of Reinforcement Learning, Q-learning [33] is a fundamental model-free Reinforcement Learning. Specifically, Q-learning enables an agent to learn the optimal action-selection policy through iterative updates of action-value estimates based on experience [34]. Unlike methods that require a model of the environment, Q-learning directly approximates the value of actions, making it suitable for a wide range of applications where the environment dynamics are unknown.

In more detail, Q-learning is the optimal action-value function, denoted q_*, which represents the maximum expected rewards achievable from each state-action pair under the optimal policy. This optimal action-value function is the cornerstone for deriving optimal policies, as they share the same q_*. The optimal policy can be defined in terms of q_* as follows [33]:

$$q_*(s, a) = \max_{\pi} q_\pi(s, a), \ \forall \ s \in S \text{ and } a \in A(s) \tag{6}$$

For the state-action pair (s, a), this function represents the expected return for selecting action a in state s, followed by adhering to an optimal policy. Hence, q_* can be formulated in relation to optimal state-value function v_*, as follows [33]:

$$q_*(s, a) = \mathbb{E}[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a] \tag{7}$$

Because v_* is the optimal value function, it can be written in a special form without referencing any specific policy. This represents the Bellman equation [35] for v_* or the Bellman optimality equation. Intuitively, the Bellman optimality equation expresses the fact that the value of a state under an optimal policy must equal the expected return for the best action from that state [35, 36]:

$$
\begin{aligned}
v_*(s) &= \max_{a \in A(s)} q_*(s,a) \\
&= \max_a \mathbb{E}_{\pi_*} \left[G_t \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi_*} \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E}_{\pi_*} \left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \mid S_t = s, A_t = a \right] \\
&= \max_a \mathbb{E} \left[R_{t+1} + \gamma v_*(S_{t+1}) \mid S_t = s, A_t = a \right] \\
&= \max_{a \in A(s)} \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma v_*(s') \right]
\end{aligned} \tag{8}
$$

The Bellman optimality equation for the action-value function q_* is [35, 36]:

$$
\begin{aligned}
q_*(s,a) &= \mathbb{E} \left[R_{t+1} + \gamma \max_{a'} q_*(S_{t+1}, a') \mid S_t = s, A_t = a \right] \\
&= \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma \max_{a'} q_*(s', a') \right]
\end{aligned} \tag{9}
$$

In conclusion, the Bellman optimality equation provides a unique solution for the optimal value function v_*, independent of the policy. This equation forms a system of equations, one for each state, that can be solved using methods for nonlinear systems [28]. Once v_* is found, an optimal policy can be determined by selecting actions that maximize the value function q_*. A "greedy" policy concerning v_* is optimal because the optimal state-value function already considers long-term rewards. Alternatively, the optimal action-value function q_* can be used to select actions more easily by directly maximizing $q_*(s,a)$ for each state s, eliminating the need to consider future states or environment dynamics.

2.1.2 Deep Q-learning

As described, traditional Q-learning algorithms update value estimates using a tabular approach. This method becomes impractical for environments with large or continuous state spaces due to the exponential growth of the state-action space. To address this limitation, Deep Q-learning employs deep neural networks as the function approximation algorithm [37] to estimate the optimal action-value function $q_*(s,a)$. This integration of deep learning with Reinforcement Learning enables agents to learn optimal policies directly from high-dimensional sensory inputs and complex tasks [31].

In Deep Q-learning, the action-value function $Q(s,a,\theta_i)$ is parameterized by a set of weights θ_i of the Q-network at iteration i, commonly referred to as a DQN [31]. The objective is to minimize the difference between the predicted and target Q-values

derived from the Bellman optimality equation [28]. To perform experience replay, the agent's experiences $e_t = (s_t, a_t, r_t, s_{t+1})$ are stored at each time-step t in a set D_t defined as $U(D) = [e_1, \ldots, e_t]$. During learning, Q-learning updates are applied on minibatches of experience $E(s, a, r, s') \sim U(D)$ drawn uniformly randomly from the pool of stored samples D_t. The Q-learning update at iteration i uses the following loss function [31]:

$$L_i(\theta_i) = \mathbb{E}_r \left[(y_i - Q(s, a, \theta_i))^2 \right] \tag{10}$$

where:

- E_r denotes the experience $E(s, a, r, s') \sim U(D)$.
- $y_i = r + \gamma \max_{a'} Q(s', a', \theta_i^-)$
- r is the reward.
- $\gamma \in [0, 1]$ is the discount rate.
- θ_i are the parameters of the Q-network at iteration i.
- θ_i^- are the network parameters used to compute the target Q-network at iteration i.

The value $Q(s, a, \theta_i)$ represents the Q-value estimated by the current Q-network with parameters θ_i for the current state-action pair (s, a). It is the model's prediction of the expected cumulative reward starting from state s, taking action a, and after that following the current policy.

On the other hand, the expression y_i calculates the value that the Q-network aims to achieve during training. In this context, r represents the immediate reward received after the agent takes action a in state s. The term γ is the discount factor that adjusts future rewards to their present value, ensuring immediate rewards are weighted more heavily than distant future rewards. The component $\max_{a'} Q(s', a', \theta_i^-)$ denotes the maximum predicted Q-value for the next state s', evaluated over all possible actions a'. This maximum is estimated using the target network with parameters θ_i^-, a stabilized version of the Q-network. By combining the immediate reward and the discounted estimate of the best possible future rewards, the target Q-value y_i provides a benchmark for updating the Q-network's parameters to minimize the loss function and improve policy performance.

In summary, Deep Q-learning extends traditional Q-learning by incorporating deep neural networks to approximate the action-value function, making it feasible to tackle complex environments with high-dimensional state spaces. This approach has opened new avenues in Reinforcement Learning, allowing agents to learn directly from raw sensory data and enabling previously intractable applications with classical methods.

2.2 5G Core Network

5G, or the fifth generation of mobile networks [1], is designed to meet the growing demand for faster, more reliable, and more efficient wireless connectivity across various industries and applications [38]. 5G networks provide high-performance capabilities compared to their predecessors [39]. They offer higher data rates, lower latency, and greater capacity [40]. These features support advanced use cases such as autonomous vehicles, virtual reality, augmented reality, smart cities, and the Internet of Things [38].

The 5G core network operates based on a slice architecture that uses network virtualization technology to create multiple logical networks, or slices, on a single physical infrastructure [41]. Each network slice is configured with specific network functions to efficiently fulfill the characteristics of a particular service type. The core network functions, such as AMF, SMF, and UPF, are designed in a modular way, enabling flexible network slicing [42]. The AMF is responsible for handling signaling, mobility management, and connection management between the User Equipment (UE) and the core network [43]. It interacts closely with the SMF to establish and manage user sessions, forwarding relevant information regarding user mobility and registration status [44].

In summary, the 5G Core comprises various Network Functions, each of which performs a specific role in the operation of the network [45]. The UPF is a critical component that handles data forwarding and traffic routing between the Radio Access Network (RAN) and external data networks [46]. The SMF plays a central role in managing sessions, enforcing policies, and dynamically assigning data paths in coordination with the UPF [47]. The SMF and UPF communication ensures efficient data handling, optimized routing paths, and effective session continuity. With the separation of the control plane (AMF, SMF) and the user data plane UPF, 5G achieves improved efficiency, scalability, and performance [42]. Network slice selection is performed during UE registration [48] and the UE can seamlessly connect to different slices, ensuring efficient resource utilization and flexible service deployment across different geographic locations.

2.2.1 Session Management Function (SMF)

The SMF holds a critical position in 5G networks due to its extensive range of responsibilities. It establishes, modifies, and terminates sessions for UE, ensuring seamless connectivity [49]. The SMF allocates IP addresses and manages dynamic session parameters, facilitating efficient communication [50]. It coordinates with the UPF to establish user data paths and configures traffic rules for the UPF [45]. The SMF enforces QoS policies and ensures session continuity during mobility events, providing a consistent user experience [51].

Additionally, the SMF supports network slicing by managing session information per-slice basis, allowing for tailored services within the network [52]. It provides DHCP services, conducts lawful interception of session management events, handles charging processes, and supports roaming capabilities. The SMF exposes service operations that enable the management of PDU sessions [53], control of policy settings, traffic rules, and subscription to notifications on session management events.

2.2.2 Access and Mobility Management Function (AMF)

The AMF is primarily responsible for managing UE access and mobility [54]. It handles registration, connection, reachability, mobility management, and access authentication functions [55]. By interfacing with the RAN and other core network functions, the AMF ensures seamless mobility and connectivity for users, facilitating efficient handovers and session continuity across the 5G network [56].

The AMF decouples access-specific functions from session management, allowing for a more flexible and scalable network design. This separation enables the SMF to focus on session-related tasks [57] while the AMF manages access and mobility procedures [58]. Such architectural modularity enhances the network's ability to support diverse services and massive connectivity requirements inherent in 5G deployments.

Moreover, the AMF utilizes the Stream Control Transmission Protocol (SCTP) [59] for reliable transport of signaling messages between the next-generation NodeB (gNodeB) and the core network [60]. SCTP serves as the underlying transport protocol for the NG Application Protocol (NGAP), which is used for signaling between the RAN and the AMF [54]. Using SCTP enhances the robustness and reliability of signaling procedures, ensuring that critical control messages are delivered in order and without loss.

2.2.3 User Plane Function (UPF)

The UPF is a pivotal component of the 5GC network architecture, responsible for handling user data traffic in order to minimize the latency of the control plane [61]. It performs crucial functions such as packet routing and forwarding, QoS enforcement in coordination with the SMF, traffic inspection, latency optimization, and session relocation avoidance [62]. The UPF acts as the anchor point for mobility, enabling seamless handovers and session continuity for mobile users.

The UPF utilizes the User Datagram Protocol (UDP) as part of the GPRS Tunneling Protocol [63] for the User Plane (GTP-U), which is essential for encapsulating and tunneling user data packets over the 5G network [64]. UDP provides a lightweight transport layer protocol that, when combined with GTP-U, allows for efficient and low-latency transmission of user data without the overhead of connection-oriented protocols like Transmission Control Protocol (TCP) [65]. This utilization of UDP in the UPF enhances data transmission performance in 5G networks, meeting the stringent requirements for high throughput and low latency.

3 Network Exposure Considerations

We assume GORGO is applied when attackers gain access to an internal infrastructure component of the target 5G+ network. Such access may result from exploiting misconfigured access controls or other vulnerabilities compromising the network perimeter. This scenario allows GORGO to model attack strategies more effectively. The attackers' ability to leverage GORGO depends on their foothold within the internal network, enabling them to exploit weaknesses and conduct targeted disruptions. This reflects realistic threat scenarios where attackers have infiltrated the internal domain to initiate DoS attacks.

However, this assumption is nullified if any of the network services are directly exposed to the Internet without appropriate security controls. Within these restrictions, the attackers can bypass traditional internal access controls, enabling them to apply GORGO without relying on prior compromise of internal components. In such a case, the attackers do not need to gain unauthorized access to the internal components

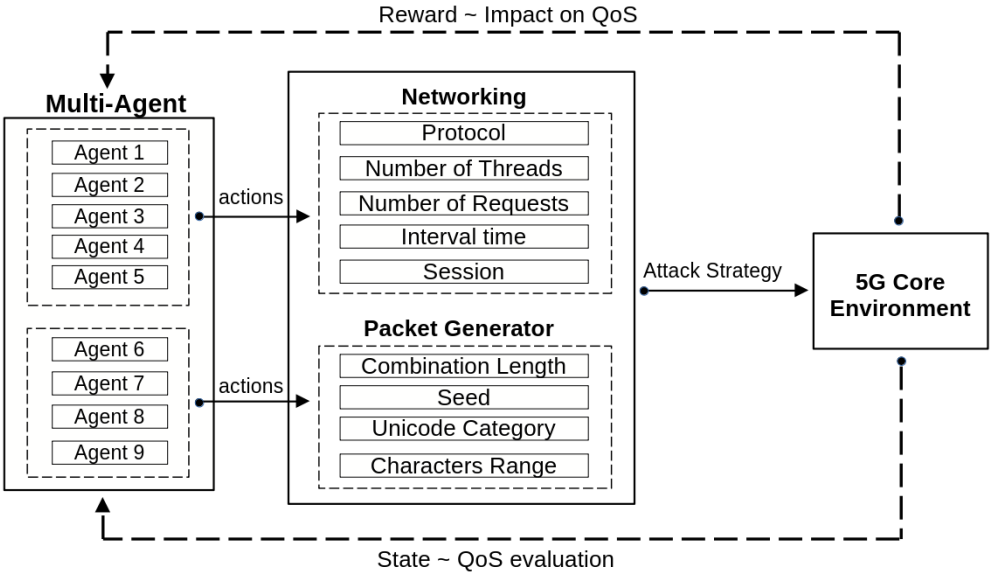

Fig. 1: Operational flow of GORGO's interaction and adaptive strategy in a 5GC environment.

to apply GORGO. The presence of publicly accessible interfaces simplifies the execution of network attacks, and GORGO's capabilities can be utilized to uncover and exploit vulnerabilities in these exposed services, thus underscoring the importance of securing both internal infrastructure and public-facing components of 5G+ networks.

4 GORGO Deep Q-Learning Neural Network

This Section details how GORGO interacts with the 5G+ environment and adjusts its strategies to successfully execute DoS attacks. It delves into the approach GORGO uses to assess the quality of the targeted services. The assessment outlines how the rewards it receives are determined by the extent of the disruption it causes. Additionally, it examines the role of each autonomous agent and the individual decisions they are required to make. The Section concludes with an in-depth exploration of the neural network architecture employed by each agent.

4.1 Overview

GORGO utilizes Reinforcement Learning to assess the resilience of 5G+ infrastructures against advanced DoS attacks driven by AI. With multiple Deep Q-Learning agents working together, GORGO formulates highly effective attack strategies, making it the only tool capable of such comprehensive evaluations. It targets real 5GC infrastructures using minimal computational resources to cause maximum disruption, and provides detailed insights into QoS degradation, helping identify the most impactful attack strategies.

GORGO is capable of attacking four protocols TCP, UDP, SCTP and HTTP, either individually or in combination and also generates custom packets for proprietary fuzzing methods, proving highly effective at exposing vulnerabilities. Its efficiency is demonstrated by its ability to detect security gaps missed by traditional penetration testing tools. In addition to its evaluation capabilities, GORGO generates detailed reports on the attack structure and impact, offering valuable insights into latency, throughput, and packet loss, making it an essential tool for network administrators aiming to strengthen system resilience and provide the corresponding network data to train AI-based IDS.

The following paragraphs present the implementation of GORGO in 5GC network, as well as the flow of the entire evaluation process. Initially, the operator specifies the desired IP address to be scanned. Subsequently, a comprehensive scan is conducted across all ports to ascertain the usage of TCP/HTTP, UDP and SCTP protocols. This process leverages sockets and concurrent threading to expedite the scanning duration. The socket function dispatches a request and awaits a response. The receipt of a response indicates that the port is operational and identifies the service in operation, otherwise the port is considered closed. The threading module is employed to parallelize the process, thereby accelerating the scanning phase.

Following the identification of open ports and their respective protocols, the QoS is assessed using parallel threads. This assessment involves evaluating latency, throughput, and packet loss, as will be outlined in Section 4.2. The initial state for the agents of the DQN is then set based on these evaluations. Although each agent starts from the same state, they generate unique actions, which are detailed in Section 4.4. The combination of these actions constitutes the final attack strategy.

The evaluation of service quality is continually updated with each round. A round is defined by the execution of an attack strategy, influenced by the agents' actions. Figure 1 illustrates the procedure through which GORGO interacts with the 5G+ environment and the actions made by each agent based on the QoS state. According to these actions it receives the reward which is common to all agents. With this methodology, agents can be trained autonomously, but learn to cooperate through the common reward they share. Further information on the reward process will be outlined in Section 4.3.

GORGO underwent training through a total of 1000 episodes, engaging with its environment in each 5G+ component. An episode is deemed complete either after 10 rounds are finished or when a total disruption of service is accomplished. If GORGO is unable to achieve a DoS within these 10 rounds, the process resets and advances to the next episode.

It is essential to highlight that service quality evaluations are conducted after each round, rather than only at the end of each episode. This approach aims to accurately reflect the real-time status of the target-service. Given the nature of 5G+ infrastructure, network traffic may naturally escalate over time due to regular user activity, independent of any attack strategy. This method ensures that the learning process for GORGO is precise, with the final reward it earns being directly linked to its unique impact on service degradation, and not influenced by the incidental effects of regular user activity.

4.2 Quality of Service evaluation

Before GORGO initiates an attack strategy, it conducts a QoS evaluation to understand the normal operational conditions without malicious disruptions. This assessment aims to create a baseline by observing the QoS in a non-compromised state, which can later be compared to the QoS post-attack. This comparison is crucial in shaping GORGO's training methodology, as it highlights the impact of the attack on service quality. By analyzing the differences in QoS before and after the attack, it becomes possible to determine the extent to which the attack degraded the infrastructure's quality or whether it remained unaffected.

GORGO's objective is to significantly deteriorate the QoS, rendering the infrastructure unable to handle additional requests from regular users, thereby causing a DoS and total collapse of the 5G infrastructure. The metrics taken into account for the QoS assessment [66] are latency [67], throughput [68], and packet loss [69]. Specifically, the evaluation takes into account the following parameters:

QoS evaluation on Latency

- Min (Minimum Latency): The minimum round-trip time in milliseconds (ms) for the packet to reach the host and for the response to return to the sender.
- Avg (Average Latency): The average round-trip time in milliseconds.
- Max (Maximum Latency): The maximum round-trip time in milliseconds.
- Mdev (Mean Deviation of Latency): The mean deviation of the round-trip times in milliseconds.
- Packet Loss: The percentage of packets that are lost in transit and do not reach their intended destination.

QoS evaluation on Throughput

- Min (Minimum Throughput): The lowest data transfer rate recorded during a test period. It represents the slowest speed at which data was transferred between two devices on the network.
- Avg (Average Throughput): The mean data transfer rate over a test period. It gives a general idea of the network performance and is more stable as a metric since it is less affected by short- term variations.
- Max (Maximum Throughput): The highest data transfer rate recorded during the test. Maximum Throughput shows the peak performance of the network, which might only be sustainable for short periods.
- Mdev (Mean Deviation of Throughput): Indicates the variability of the throughput measurements around the mean. A low mdev would suggest consistent throughput, while a high mdev would indicate that the transfer rates fluctuated significantly during the test period.

4.3 Reward function

During the DoS attack, GORGO evaluates in parallel the QoS and at the end of the attack, it calculates the change in service quality for each port individually and

collectively. This real-time evaluation is key to understanding how different aspects of the network service are being affected as the attack unfolds. The focus is on quantifying the impact of the attack in terms of percentage changes in latency, throughput, and packet loss for each service. This analysis is not just limited to individual services or ports but also encompasses an overall assessment of the network's performance.

The core of this methodology is the evaluation method GORGO is using, which orchestrates concurrent measurements across various network protocols including TCP, UDP, SCTP and HTTP based on which of them the target service is using. This is achieved by spawning multiple threads, each dedicated to a specific protocol. These threads independently gather data on key parameters such as latency, throughput, and packet loss as previously mentioned.

Once the attack phase concludes, GORGO steps in to process and analyze the aggregated data. It calculates the percentage change in QoS psarameters from the initial state (pre-attack) to the final state (post-attack). A unique aspect of this analysis is the handling of a special case where 100% packet loss and 0 (ms) latency in the final state are interpreted as maximum impact, because this will mean that the target has been rendered inoperable by the DoS attack.

The methodology reaches its peak with the calculation of a final reward value, which is derived from a method that aggregates the changes in the average values of Latency, Throughput and Packet Loss. The Equation 11 calculates the percentage change between two values, where X_{before} represents the pre-attack state and X_{after} represents the post-attack state.

$$\Delta X = \begin{cases} \frac{X_{\text{after}} - X_{\text{before}}}{X_{\text{before}}} \times 100, & \text{if } X_{\text{before}} \neq 0 \\ X_{\text{after}}, & \text{if } X_{\text{before}} = 0 \text{ and } X_{\text{after}} \neq 0 \\ 0, & \text{if } X_{\text{before}} = 0 \text{ and } X_{\text{after}} = 0 \end{cases} \tag{11}$$

The reward is then derived from the sum of the percentage changes in the average values of Latency, Throughput, and Packet Loss, given by:

$$QoS_{\text{R}} = \begin{cases} \Delta X_{\text{Latency}} + |\Delta X_{\text{Throughput}}| + \Delta X_{\text{Packet Loss}}, & \text{if } \Delta X_{\text{Throughput}} < 0 \\ \Delta X_{\text{Latency}} - \Delta X_{\text{Throughput}} + \Delta X_{\text{Packet Loss}}, & \text{if } \Delta X_{\text{Throughput}} > 0 \end{cases} \tag{12}$$

where:

- $\Delta X_{\text{latency}}$ is the percentage change in latency.
- $\Delta X_{\text{throughput}}$ is the percentage change in throughput.
- $\Delta X_{\text{packet loss}}$ is the percentage change in packet loss.

The calculated value is then normalized to provide a comprehensive measure, encapsulating the overall impact of the attack on the network's Quality of Service. This single metric effectively summarizes the extent of the disruption caused by GORGO's attack. For example, Table 1 shows measurements referring to the quality of the SCTP service on AMF at port 38412, before and after GORGO's attack.

Table 1: SCTP port 38412 QoS evaluation results before and after attack

Metrics	Before Attack	After Attack
Min Latency (ms)	0.10	0.09
Avg Latency (ms)	0.50	2.05
Max Latency (ms)	4.34	261.78
Mdev Latency (ms)	0.45	10.40
Packet Loss (%)	0	3.62
Min Throughput (byte/s)	1.91	0.03
Avg Throughput (byte/s)	31.95	24.41
Max Throughput (byte/s)	81.92	81.92
Mdev Throughput (byte/s)	21.66	22.72

It is crucial to understand how the sign (positive or negative) is assigned to each value, based on whether there is an increase or decrease from the baseline. For Latency and Packet Loss, an increase from the initial values results in a positive sign for the calculated reward. Conversely, a decrease in these metrics would lead to a negative value. With Throughput, the logic is inverted, a decrease relative to the pre-attack measurement is considered positive because a drop in Throughput signifies a reduction in service quality. On the other hand, an increase in Throughput would result in a negative contribution to the reward. This approach reflects the fact that a lower Throughput and higher Latency or Packet Loss typically indicate a worsening in service quality which is the aim of GORGO.

The comparison of the example above clearly indicates a decline in Quality of Service. In detail, there was a substantial increase in Latency by about +310%, a decrease in Throughput by around +23.60%, and an increase in Packet Loss by approximately +3.62%. These measurements are used to compute the QoS_{R}, which is the aggregate of these three values, amounting to +337.22. However, the final reward is shaped by a normalized perspective as seen in the following equation:

$$Normalized_{\mathrm{Reward}} = \frac{QoS_{\mathrm{R}}}{MaximumPossibleChange} \times 100 \tag{13}$$

The $MaximumPossibleChange$ for each metric is hypothetically set to 1000 as this value means that there is a complete collapse of the service. In this case then the reward is equal to the value of the $MaximumPossibleChange$, as the intention is to learn to cause a total denial of service by giving a large reward when it succeeds. While in the opposite case, normalizing the total reward would involve dividing 337.22 by this $MaximumPossibleChange$ and multiplying by 100. Through this method, the $Normalized_{\mathrm{Reward}}$ value would be about $33.72 \simeq 34$. This value offers a standardized measure of overall impact, simplifying comparisons and interpretations across different scenarios or assessments.

4.4 Autonomous agents actions

GORGO consists of 9 autonomous agents, each with a specialized role. The initial agent is responsible for choosing the port to attack. This is achieved by first conducting

a port scan to identify open ports. Concurrently, this agent also identifies which ports correspond to specific protocols such as TCP, UDP and SCTP. As a result, the total number of actions that this first agent can perform corresponds directly to the total number of ports that are available.

The role of the second agent in GORGO is to determine the total number of threads to be deployed for the DoS attack. To evaluate GORGO's effectiveness under resource constraints, the upper limit for threads has been capped at 200. The minimum number of threads that can be utilized is one, with the thread count increasing in increments of 10 up to the maximum The role of the second agent in GORGO is to determine the total number of threads to be deployed for the DoS attack. To evaluate GORGO's effectiveness under resource constraints, the upper limit for threads has been capped at 200. The minimum number of threads that can be utilized is one, with the thread count increasing in increments of 10 up to the maximum limit. Consequently, the second agent has 21 (1, 10, 20, ..., 200) different options for the number of threads it can choose to deploy.

Similarly, the third agent is tasked with determining the number of requests each thread will execute. The minimum number of requests per thread is established at 1, and the maximum is set at 500, increasing in increments of 10. This results in a total of 51 available options (1, 10, 20 ..., 500) for the number of requests.

The fourth agent's responsibility is to select the interval in seconds between requests issued by each thread, offering four choices: 0, 1, 2 or 3 seconds. Meanwhile, the fifth agent is charged with deciding whether to maintain the request session active or to close it immediately after a request is made. Therefore, the available options for the fifth agent are a total of 2.

The remaining 4 agents are tasked with setting parameters for the custom made packet generator. Based on their decisions, the characteristics such as the type, architecture, quantity, and the specific combination of packets to be generated and sent to the target are configured.

The packet generator was developed to take 5 parameters as input. The first parameter is the number of packets to be created and is equal to the number of requests for each thread.

The second parameter concerns the length of the combinations which specifies how many characters will be combined together to form a single payload. For example, if combination length is set to 2, each generated combination will be a sequence of two characters. This means if the available set of characters is [A, B, C] and combination length is 2, the generated combinations will be (A, A), (A, B), (A, C), (B, A), (B, B), (B, C), (C, A), (C, B), (C, C). The sixth agent can choose the number of combinations, with the limit set at 5000. This agent has a range of choices starting at 0 and ending at 5000, with increments of 100 (0, 100, ..., 4900, 5000). In total, there are 51 different options available for the agent to select from. Combination length of 0 means no packets will be generated and therefore no fuzzing will occur. In fuzz testing the impact of this approach is significant because it defines the complexity and depth of the test cases. Longer combinations can test the target system's handling of more complex and varied inputs.

Table 2: Unicode characters categories

Category	Description
1. Cc	Control Characters
2. Cf	Format Characters
3. Cs	Surrogate Characters
4. Co	Private Use Characters
5. Cn	Unassigned Characters
6. So	Other Symbols
7. Sc	Currency Symbols
8. Sk	Modifier Symbols
9. Sm	Mathematical Symbols
10. Me	Enclosing Marks
11. Mn	Non-spacing Marks
12. Mc	Spacing Combining Marks
13. Lm	Modifier Letters
14. Zs	Space Separators
15. Zl	Line Separators
16. Zp	Paragraph Separators

Table 3: Unicode characters ranges

Range	Description
1. 0x0000-0x001F	Text processing control characters
2. 0x0080-0x009F	Extended ASCII character sets
3. 0xD800-0xDFFF	UTF-16 characters outside the BMP
4. 0x2000-0x206F	Punctuation marks
5. 0x20D0-0x20FF	Diacritical marks for symbols
6. 0xE000-0xF8FF	Custom characters
7. 0xFE00-0xFE0F	Visual styles for Unicode characters
8. 0xFFF0-0xFFFF	Special purpose characters
9. 0x0700-0x077F	Syriac script characters
10. 0x2C00-0x2C5F	Oldest known Slavic alphabet

The third parameter of the packet generator is about the seed which is used to initialize the random number generator. Setting a seed ensures that the sequence of random numbers (and hence the selection of characters) is repeatable and consistent across different runs. This is crucial for reproducing tests or debugging, as it allows the same sequence of characters to be generated every time the program is run with the same seed. The seventh agent is responsible for choosing the seed value. It has 101 different options with the minimum value being 0 and the maximum being 100.

The fourth parameter is related with the index determine which category of Unicode characters the generator will focus on when generating test data for fuzzing. The list of Unicode characters is presented in Table 2. There are 16 categories each of which represents a different group of Unicode characters, classified according to their general properties and usage. The eighth agent chooses the index from Table 2 which acts as a pointer to a specific category within the list. For example, if the agent chooses the index to be 0, the generator will select the first category in the list, if it selects the index to be 1, it will select the second category, and so on. During the fuzzing character generation process, the method iterates over all Unicode code points and checks the category of each character. If the character's category matches the one pointed to by index, the character is considered for inclusion in the fuzzing test data. This selection mechanism based on Unicode characters allows the fuzzer to focus on specific types of characters. By altering this index, the generator can be directed to generate test data with different character sets, each potentially triggering different behaviors or vulnerabilities in the system being tested. This approach enhances the thoroughness and effectiveness of the fuzz testing by ensuring a diverse range of input character types.

The final parameter of the packet generator class plays a critical role in determining the specific Unicode character ranges. Each tuple in the Table 3 represents a specific range of Unicode characters. For example, a tuple (0x0000, 0x001F) defines a range that includes Unicode characters from the code point 0x0000 up to and including 0x001F. These ranges are selected based on their relevance to the fuzzing process. They often include characters that are likely to produce interesting behaviors in systems, such as control characters, private use characters, or special symbols. This process is

separate from the selection of characters based on Unicode characters categories as indicated by the Table 2. It focuses instead on specific character sequences that might not be captured by category alone. The role of the ninth agent is to choose one of these 10 ranges presented in Table 3. For example if the agent choose the 0 option then the range is configured for 0x0000-0x001F, if it choose option 1 the range is configured for 0x0080- 0x009F and so on.

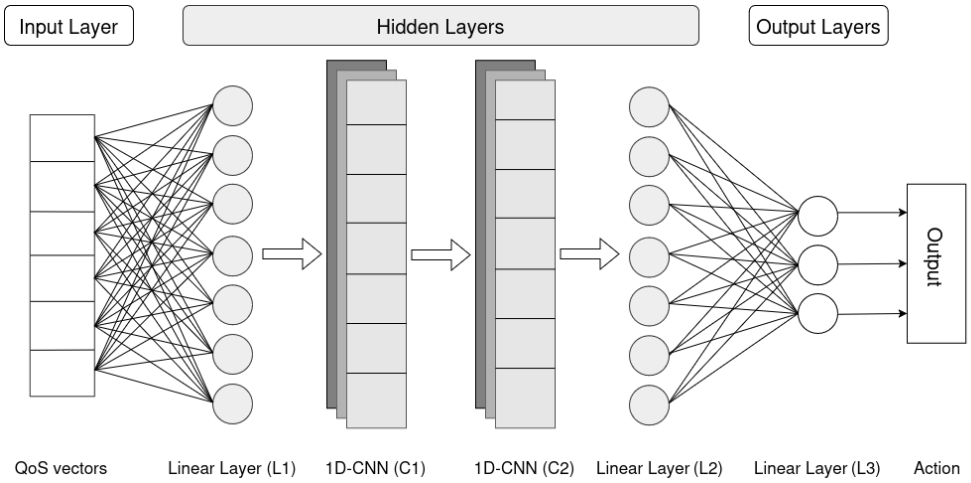

Fig. 2: Agent's neural network architecture.

4.5 Agent's neural network architecture

Each agent described utilizes its own DQN architecture, with around 6.3 million trainable parameters per agent. Consequently, GORGO encompasses approximately 57 million learnable parameters in total.

The neural network architecture was specifically designed for Reinforcement Learning applications and it is notable for its combination of convolutional and linear layers, making it particularly effective for high- dimensional observation spaces. Figure 2 illustrates this architecture, in which L1, L2, and L3 represent the linear layers, whereas C1 and C2 represent the convolutional layers. The initial part of the network consists of a fully connected linear layer that expands the input tensor's dimensionality to 512 features. This expansion prepares the input for more complex processing in subsequent layers.

Following this initial expansion, the network utilizes two convolutional layers, characterized by a kernel size of 1 and a stride of 2. The first convolutional layer increases the channel size from 512 to 1024, while the second maintains it at 1024. This design facilitates a deep and nuanced feature extraction from the input data, which is vital for understanding the environmental state patterns.

After the convolutional processing, the network architecture includes a fully connected layer that downscales the feature dimensions from 1024 to 512. This is followed

by a key linear layer, which maps these 512 features to the action space. This mapping is essential as it outputs the action values or advantages for each possible action in a given state. It must be noted that each autonomous agent has a unique action space tailored to the specific task it is designed to perform. Therefore each agent receives as input the same information but produces different outputs.

One of the unique features of this network is its ability to handle input tensors of varying lengths, with a maximum tensor length set to 2048. This flexibility is crucial in Reinforcement Learning scenarios where state representations may vary in size. The network adeptly manages this by truncating or padding tensors to ensure consistent input size for processing. During the forward pass, the network processes the input state through the linear layer, followed by the convolutional layers and finally through the fully connected layers. The output is the action value, crucial for decision-making in Reinforcement Learning.

The neural network is optimized using the Adam optimizer, a widely used optimizer in Deep Learning and employs a Mean Squared Error (MSE) loss function. Also, the Rectified Linear Unit (ReLU) activation function is used for each layer, allowing positive values to be retained while setting negative values to zero. A notable aspect of this architecture is its learning rate, set to a fine-tuned value of 1e-5, allowing for precise adjustments during training.

Moreover, the network is designed to operate on CUDA-enabled devices, if available, ensuring accelerated computation on GPUs, which is particularly beneficial for the computationally intensive training processes in Deep Reinforcement Learning. In summary this type of architecture is well-suited for mapping environmental states to action values, a critical function in the decision- making processes within Reinforcement Learning.

5 Evaluation

This section presents the training outcomes obtained in a real-world 5G environment [70] and in a B5G infrastructure [71]. We focus on the SMF, the UPF and the AMF because these three components form the indispensable backbone of every 5G-core slice. Each slice must run exactly one AMF together with at least one SMF and UPF, so disabling any of them cripples all services that share the slice. AMF and SMF operate on the control plane, whereas UPF anchors the user plane, which allows us to observe how a single attack vector can propagate across planes and amplify its impact. Each function also relies on a different transport protocol: SMF communicates over HTTP on top of TCP, UPF terminates GTP-U carried by UDP and AMF handles SCTP signaling, so targeting them allows us to test whether our Reinforcement Learning penetration-testing tool can craft effective DoS patterns against protocol families with distinct characteristics and to measure how resilient each function remains under protocol-specific stress. The remainder of this section details the training duration and methodology and examines how GORGO's attacks affect quality of service in order to determine whether 5G+ services are vulnerable to AI-driven

attacks, assess the resulting impact on service quality and analyze how such exploitation occurs, thereby providing insights into 5G+ infrastructure resilience and guidance for strengthening defense mechanisms against intelligent threats.

5.1 Testbeds

GORGO's training was conducted within the SMF, UPF, and AMF components of a 5G testbed. More specifically, it is an SDN-based 5G implementation [70] that enables real-time monitoring and traffic management among the 5G network components. The SDN controller serves as the central control point, enabling the integration of effective monitoring modules such as real-time traffic statistics gathered from the network through the OpenFlow protocol. On top of the SDN system, there is a mix of AI-based IDS and Simple statistical models for threat identification [72]. Overall, the system's implementation is based on the integration of two key components: (i) Free5GC [73], which offers a cloud-native architecture for 5G networks and supports Non-Standalone (NSA) deployments anchored in LTE, and (ii) the SDN framework presented in work [74], which integrates SDN-based networks with containerized microservices.

Following its initial training, GORGO evaluated the resilience of a B5G experimentation platform [71], which is based on Open5GS [75]. This B5G environment is an experimental, cloud-native extension of a Standalone (SA) 5G core designed to prototype services and architectures that will be utilized by future 6G standards. This advanced environment employs Kubernetes for orchestrating core network components, where each network function operates within individual Kubernetes pods interconnected through defined sequences. The infrastructure also integrates a comprehensive RAN, leveraging Open-RAN (O-RAN) solutions alongside advanced B5G core solutions. This setup provided GORGO with a robust technological foundation for assessing and enhancing network resilience in next-generation mobile networks.

Knowledge transfer is organized from SMF to UPF and finally to AMF so that the agent progresses through the increasing complexity and control hierarchy of the 5G core. Beginning with the SMF allows GORGO to learn in a structured, deterministic setting centered on session control, QoS enforcement and session management, giving the agent a stable grasp of session logic. The learned parameters are then transferred to the UPF component, where those session-level insights help the agent understand real-time packet forwarding over GTP-U and recognize patterns that saturate forwarding resources. At this stage, the UPF also forces GORGO to deal with stateless UDP flows, teaching it how to establish, refresh and expire lightweight connectionless tunnels so it can maintain throughput guarantees and perform sophisticated UDP attacks. Finally, the agent is exposed to the AMF, whose direct interaction with UE signaling over Non-Access Stratum (NAS) introduces the greatest variability and adversarial behavior; at this stage the prior knowledge gained from SMF and UPF is used to refine the attack strategy. This bottom-up progression ensures that GORGO fully comprehends session and data-flow semantics before confronting the more dynamic and exposed control-plane function; thereby improving both convergence speed and attack effectiveness.

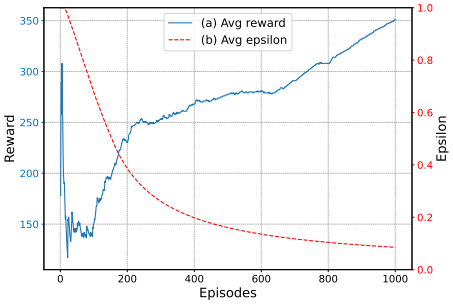

Fig. 3: Progression of GORGO's learning across episodes on 5G SMF

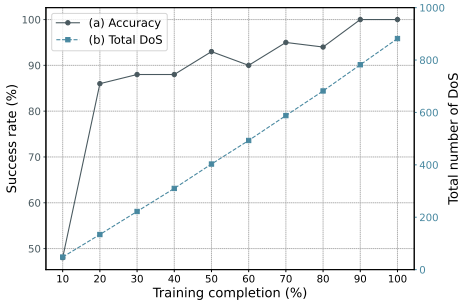

Fig. 4: Success rate of DoS per episode and total DoS attacks on 5G SMF

5.2 DoS Evaluation on SMF 5G Component

First, we present the DoS evaluation results against the SMF component within a 5G infrastructure to test the resilience against AI-DoS attacks. The deployed SMF uses two TCP ports (3868, 5868) and one HTTP port (8080). The training involved 1000 episodes, completed in 2 days, 21 hours, 56 minutes, and 15 seconds. Figure 3 illustrates GORGO's progress (line a) over the episodes and the gradual reduction of the ϵ factor (line b), demonstrating its influence on the training process. As noted, each episode concludes upon causing a DoS or after ten rounds. Thus, episodes may include multiple rounds with varying rewards, leading to an average reward per episode. Therefore, we focus on the episode's average reward rather than individual round rewards. We use a cumulative average to monitor learning progress, offering an overview of performance trends by averaging rewards from the start to the latest episode.

Figure 3 shows that as ϵ (epsilon) decreases, GORGO's rewards increase. When ϵ is high (exploration phase), choices are made randomly as the AI experiments with various strategies. As ϵ lowers (exploitation phase), GORGO makes more deliberate decisions, leading to consistent reward improvement. In summary the ϵ factor decides how often the DQN agent tries something completely random rather than using what it already knows. This randomness helps GORGO to explore different possibilities so it does not get stuck always choosing the same action. Over time, epsilon typically gets smaller so the agent does fewer random actions and uses its learned strategy. The reduction of ϵ follows the equation:

$$\varepsilon = \begin{cases} \varepsilon - 1e^{-3}, & \text{if } \varepsilon > 0.01 \\ 0.01, & \text{if } \varepsilon \leq 0.01 \end{cases} \tag{14}$$

The equation suggests that ϵ reduces linearly; however, the line (b) in Figure 3 shows an exponential decline. This discrepancy arises because, while ϵ decreases linearly within each round, the average ϵ per episode (comprising several rounds) exhibits an exponential decrease—averaging the linearly decreasing ϵ values over an episode results in an exponential trend. The exploration phase continues until episode 60, with ϵ around 0.8, leading to mostly random decisions and reward fluctuations (line a). After episode 150, GORGO makes more deliberate decisions as ϵ decreases further (exploitation phase). Rewards grow logarithmically between episodes 200 and 650, transitioning to linear growth from episode 700 onwards. This progression indicates

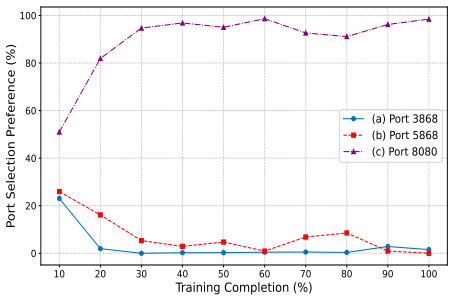

Fig. 5: Port selection preference during 5G SMF training

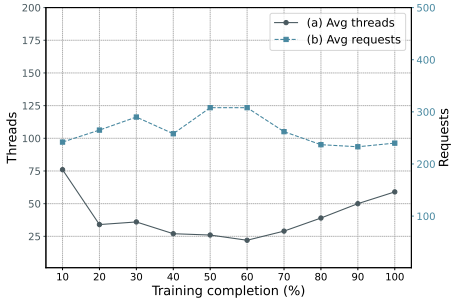

Fig. 6: Thread and request number preference during 5G SMF training

that GORGO has refined its decision-making and effectively learned to execute DoS attacks.

The frequency of successful DoS attacks on the SMF service, demonstrating GORGO's effectiveness in significantly disrupting the target, is illustrated in Figure 4. Line (a) shows the number of successful DoS attacks relative to the training completion percentage. Notably, between 90% and 100% training completion, GORGO achieved 100 successful attacks, indicating a 100% success rate in this range. Line (b) depicts the cumulative total of successful DoS attacks over the training period. Initially, up to the 10% mark, there were 48 DoS attacks. By the end of the training, the total successful DoS attacks reached 882, meaning GORGO caused DoS in 88.2% of the training episodes.

Figure 5 shows GORGO's port selection preference during training. Initially, GORGO explores all available ports with balanced usage. A strong preference for HTTP port 8080 emerges as training progresses, reaching about 95% after 30% of training and remaining the most preferred port after that. Conversely, ports 3868 and 5868 decline significantly, settling near zero by training's end. This suggests that the HTTP port 8080 is the most vulnerable, according to GORGO's decision-making.

The average number of threads and requests GORGO handles during training is shown in Figure 6. As observed, the average requests remain high and stable between 240 and 300, indicating a consistent workload. Conversely, the thread count starts high at approximately 75, decreases to around 25 at 50% training completion, and then rises to about 60 by the end.

GORGO's fuzzing category and range preferences during training are depicted in Figure 7. The heatmap displays the distribution of selected Unicode character categories and ranges; darker shades indicate higher preferences. As training progresses, GORGO increasingly prefers `Currency Symbols` and `Spacing Combining Marks`, peaking at 47% and 44% in the latter half. `Private Use Characters` are often selected early but decline over time. Ranges like `0x20D0`–`0x20FF` increase in preference, surpassing 43% later in training. `Control Characters` and `Modifier Symbols` remain consistently low, suggesting they are less effective for fuzzing. The `0x20D0`–`0x20FF` and `0x2C00`–`0x2C5F` ranges start high but decrease, while `0x0700`–`0x077F` starts low, spikes around 50% training, and becomes most preferred after 70%. Overall,

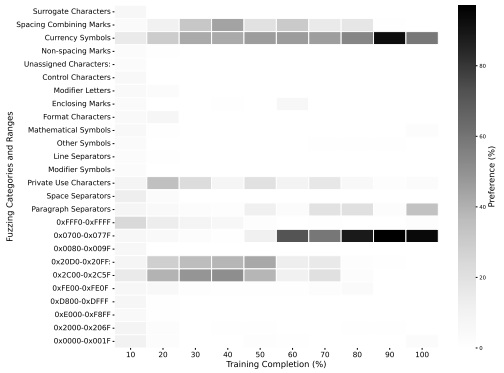

Fig. 7: Fuzzing byte categories and ranges preference during 5G SMF training

Table 4: Optimal attack strategy on 5G SMF

Strategy	**Optimal selection**
Protocol	HTTP
Port	8080
Threads	60
Requests	240
Interval (s)	2
Session	Close
Packet length	400
Seed	22
Characters	Currency Symbols
Unicode range	0x0700-0x077F

GORGO's fuzzing preferences evolve to prioritize effective categories and byte ranges for optimizing vulnerability identification in SMF component.

Table 4 presents GORGO's optimal attack strategy on SMF targeted HTTP port 8080, using 60 concurrent threads to send 240 requests at 2-second intervals for each request. The attack involved closing sessions after each request, likely to increase server-side resource consumption. Payloads were 400 bytes in length, with a seed value of 22 used to ensure consistency in the fuzzing process. The payload content included `Currency Symbols`, and characters were selected from the Unicode range `0x0700`–`0x077F`, to exploit vulnerabilities on HTTP port 8080. This strategy combined concurrency, session manipulation, and specific Unicode content to effectively maximize attack impact.

5.3 DoS Evaluation on 5G UPF Component

In the second experiment, the pre-trained weights from previous attacks on SMF were loaded and the effectiveness of GORGO against UPF was tested. This service uses the UDP protocol to communicate with the gNodeB in order to facilitate the transfer of user data between the UE and the core network. This communication is part of the Packet Data Unit session, where the UPF and gNodeB exchange user plane traffic using the UDP on port 2152. Training duration lasted 16 hours, 55 minutes, and 22 seconds. The efficiency improvement over the initial training on SMF duration is 75.80% as a result of the prior training.

Below, the evaluations of GORGO's training are presented, and its effectiveness in strategic attack selections is analyzed. The results from Figure 8 indicate an initial period of instability, where the average reward decreases before starting to recover and gradually increase. This instability is likely due to the fact that the agent has not yet fully understood the new environment, as it uses different communication protocol and is executing suboptimal actions while it adapts to the system's dynamics. However, the ϵ value remains consistently low throughout the training process, suggesting that exploration was already limited and learning relied primarily on the

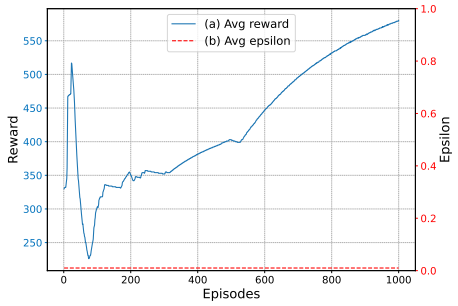

Fig. 8: Progression of GORGO's learning across episodes on 5G UPF

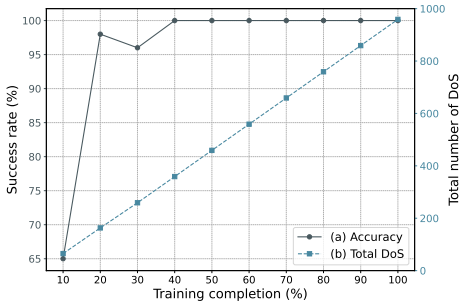

Fig. 9: Success rate of DoS per episode and total DoS attacks on 5G UPF

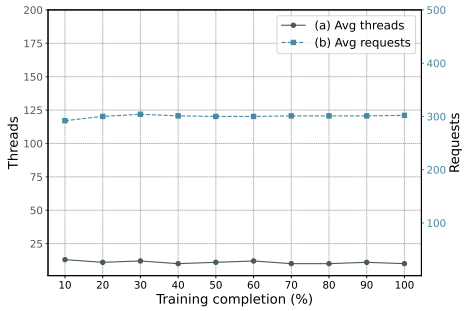

Fig. 10: Thread and request number preference during training on 5G UPF

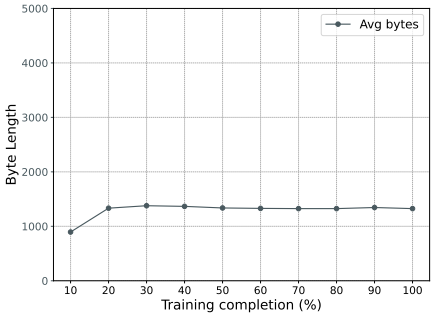

Fig. 11: Byte length selection preference during training on 5G UPF

exploitation of accumulated experiences. The steady reduction in instability and the gradual increase in reward demonstrate that the agent successfully adapts, ultimately converging toward an effective policy.

Figure 9 demonstrates a clear correlation between the completion of GORGO's training and both its attack success rate and the total number of executed DoS attacks. As training progresses, the success rate of GORGO increases significantly from an initial value of 65% at 10% training completion to nearly 100% beyond 40% completion. The most rapid improvement occurs in the early stages, where the success rate jumps from 65% to 98% between 10% and 20% completion. Although minor fluctuations are observed around 30% completion, the success rate stabilizes at 100% as training nears completion, indicating that GORGO has effectively learned to execute successful attacks with maximum efficiency. Simultaneously, the total number of performed DoS attacks follows a steady linear increase, reaching a total of 959, reflecting GORGO's growing capability to launch attacks as its training progresses. This trend suggests that with more extensive training in different environments, GORGO consistently improves its ability to execute attacks, leading to a higher volume of successful DoS strategies.

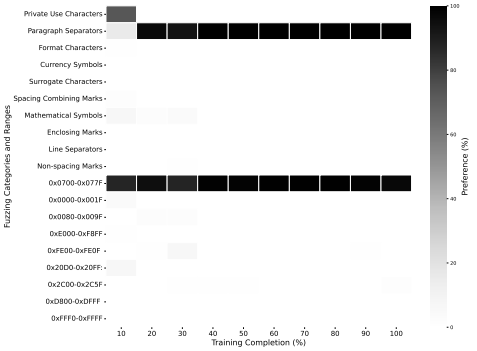

Fig. 12: Fuzzing byte categories and ranges preference during 5G UPF training

Table 5: Optimal attack strategy on 5G UPF

Strategy	**Optimal selection**
Protocol	UDP
Port	2152
Threads	10
Requests	300
Interval (s)	2
Session	False
Packet length	1300
Seed	10
Characters	Paragraph Seperators
Unicode range	0x0700-0x077F

Building on this observation, the refinement of GORGO's attack strategies can be further analyzed through its adaptive behavior in thread selection and request generation as illustrated in Figure 10. As training progresses, the average number of threads remains relatively stable, fluctuating slightly around a low value, indicating that GORGO does not significantly increase concurrency over time. The average number of requests shows a more consistent trend, stabilizing at approximately 300 requests, suggesting that GORGO maintains a steady attack rate as training completion increases. Similarly, Figure 11 shows that the average byte length of requests follows an initial growth phase, reaching a peak early in the training process before stabilizing around 1300 bytes. This behavior indicates that GORGO learns to optimize the size of fuzzing bytes rather than increasing the request number or the number of concurrent threads. The overall findings suggest that instead of scaling up attacks by increasing resource consumption, GORGO refines its attack strategy by optimizing its byte length, ensuring a balance between efficiency and effectiveness.

The optimal attack strategy for GORGO on UPF is characterized by a carefully tuned selection of parameters that maximize the reward. Table 5 presents the optimal attack strategy, which includes UDP port 2152, as it is the only active port. The number of threads remains relatively low at 10, while the request rate stabilizes at 300 requests per thread. The interval seconds between each request is set at 2 seconds. The attack does not rely on persistent session establishment, as indicated by the session parameter being set to false. The optimal packet length is determined to be 1300 bytes, aligning with previous findings that GORGO refines its attack payload size rather than increasing concurrency. The attack strategy utilizes a seed value of 10 to introduce controlled randomness, ensuring that the attack remains effective without being easily predictable. As Figure 12 illustrates fuzzing character selection is focused on `Paragraph Separators`, while they were selected from the Unicode range `0x0700`–`0x077F` to exploit vulnerabilities on UDP port 2152.

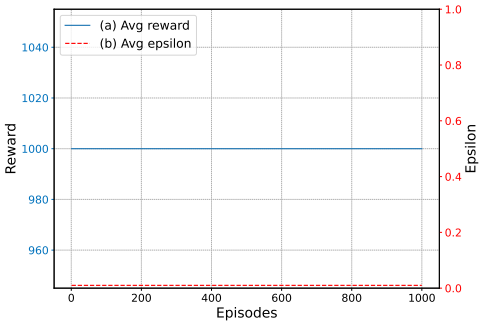

Fig. 13: Progression of GORGO's learning across episodes on 5G AMF

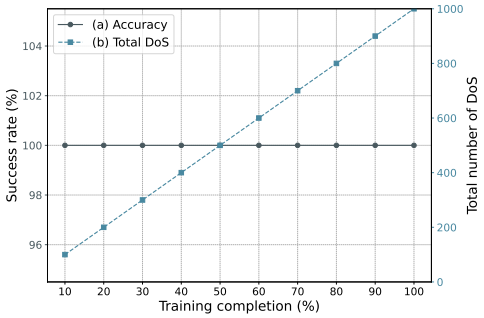

Fig. 14: Success rate of DoS per episode and total DoS attacks on 5G AMF

5.4 DoS Evaluation on AMF 5G Component

For the next experiment, we loaded the pre-trained weights from SMF and UPF attacks to test how the AMF service in 5G infrastructure responds to GORGO's attacks. The deployed AMF uses four ports; two for SCTP (36412, 38412) and two for TCP (3868, 5868). This experiment was completed in 12 hours, 56 minutes, and 54 seconds, marking an 81.49% efficiency improvement over the initial SMF training duration and 23.49% improvement over UPF training duration. This is due to the pre-trained GORGO identifying the optimal attack strategy from the very first episode, resulting in a DoS on the AMF. Thus, each episode terminated after the first round since the attacks were equally effective.

Figure 13 illustrates the rewards per episode earned by GORGO while keeping a constant, minimal ϵ (line b) to encourage the exploitation phase and enable agents to adopt learned strategies from previous training sessions against AMF. The constant ϵ ensures GORGO's choices are deliberate, successfully inducing a DoS in every episode. Line (a) in the graph depicts the rewards during the AMF training, stabilizing at 1000 (i.e., the highest reward) signifying a DoS occurrence. Figure 14 further supports these results, showing GORGO's 100% success rate against the AMF component.

GORGO's port selection preferences during AMF training are presented in Figure 15. Initially, port 36412 (SCTP) is most attacked, while port 38412 (SCTP) has the lowest preference. As training progresses, preferences for ports 3868 and 5868 (TCP) fluctuate, reflecting GORGO's exploration of optimal port choices. By mid-training, port 38412 became the most selected and port 5868 the least. Port 5868's preference peaks towards the end, while port 3868 drops sharply. This fluctuation reveals GORGO's learning process, converging on the most effective port selection strategy. Combined with the 100% success rate, this indicates that all services of AMF are vulnerable to GORGO's attacks regardless of port choice.

Figure 16 shows GORGO's thread and request preferences during AMF training. The average numbers vary slightly without a significant trend. Threads start at 80, rise to about 100 by 20% training completion, and then fluctuate around 100 with dips around 40% and 60%. Requests begin at 250, peak at 270 by 20%, and then fluctuate between 230 and 300. GORGO maintains a lower average thread count than

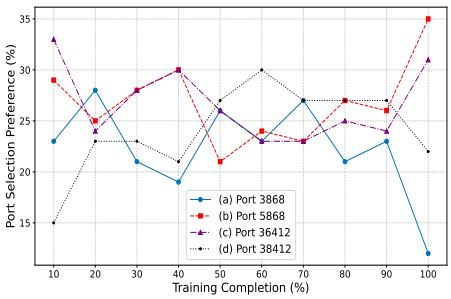

Fig. 15: Port selection preference during training on 5G AMF

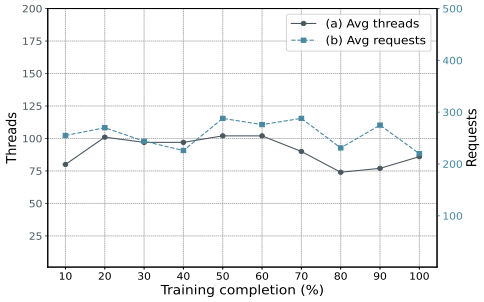

Fig. 16: Thread and request number preference during training on 5G AMF

Table 6: Optimal Attack Strategy on 5G AMF

Strategy	**36412**	**38412**	**3868**	**5868**
Protocol	SCTP	SCTP	TCP	TCP
Threads	80	130	60	90
Requests	260	200	270	340
Interval Time (s)	2	2	2	0
Session management	Keep Open	Close	Close	Keep Open
Packet Length	2900	2000	2200	2600
Seed value	9	66	94	48
Characters	Control	Format	Control	Enclosing
Unicode Range	0xD800-0xDFFF	0x2C00-0x2C5F	0x0700-0x077F	0xD800-0xDFFF

requests, indicating a conservative thread allocation that balances resource efficiency and workload without consuming excessive resources.

GORGO's optimal strategy against AMF, including a breakdown of tactics for each vulnerable port, is detailed in Table 6. It includes the number of threads, requests per thread, request intervals, session management, packet size for fuzzing, and the seed value for the packet generator. The last two elements (i.e., Characters and Unicode range) refer to Tables 2 and 3, highlighting specific choices in the packet generator used for fuzzing.

The attack patterns show that GORGO maintained a diverse strategy, distributing efforts across all four ports instead of fixating on one. Fluctuations in attack numbers per port are small, suggesting each service had similar attractiveness or vulnerability during training. Combined with a 100% success rate and a total of 1000 DoS attacks, this strongly indicates that the AMF is vulnerable to AI-enabled DoS attacks targeting both TCP and SCTP protocols, highlighting the need for enhanced security measures.

5.5 DoS Evaluation on B5G AMF Component

In the final experiment, we present the results of attacks against the AMF in the B5G infrastructure in order to uncover possible vulnerabilities. The goal is to test the resilience of B5G technology by loading pre-trained weights from GORGO, based on its experience with 5G attacks. Training lasted 20 hours, 31 minutes, and 29 seconds.

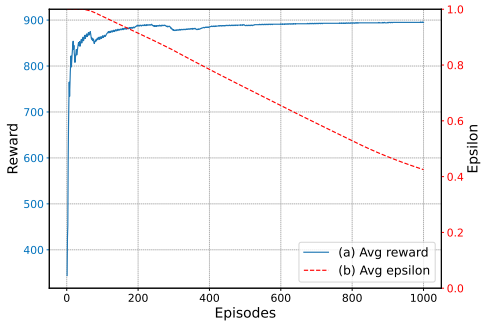

Fig. 17: Progression of GORGO's learning across episodes on B5G AMF

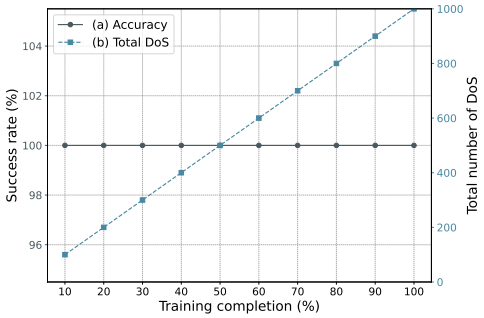

Fig. 18: DoS success rate per episode and total attacks on B5G AMF

In B5G testbed, the AMF operates solely on port 38412 using the SCTP protocol, leaving the port selection agent with only one option.

Figure 17 shows GORGO's learning progress (line a) and the decrease of the ϵ factor (line b), highlighting its training impact. Here, ϵ starts at 1 and decreases as outlined in Equation 14, encouraging exploration in this distinct setting. This approach allowed GORGO more freedom to explore, resulting in random actions until episode 90 when ϵ was highest, explaining the lower rewards observed initially. Additionally, Figure 17 illustrates the linear decrease of ϵ, contrasting with the exponential decrease in Figure 3. This is due to fewer repetition rounds needed at B5G AMF to meet episode completion. Specifically, the average repetition rounds were 4.1 at 5G SMF, 2.3 at 5G UPF, 1.0 at 5G AMF and 1.3 at B5G AMF.

The rate of successful DoS attacks on the B5G AMF, demonstrating GORGO's proficiency, is displayed in Figure 18. GORGO achieves a successful DoS attack in every episode, so line (a), which tracks DoS attacks per training completion, remains constant at 100. Line (b) shows a linear increase in total DoS attacks, with every 100 episodes yielding 100 successful attacks, resulting in a total of 1000 DoS attacks. This confirms GORGO's 100% success rate, leading to the AMF service's total collapse.

Initially, as shown in Figure 19, GORGO launched 110 – 130 threads with an average of 280 requests per thread, using 2220 bytes for fuzzing (Figure 20). In the final stages (episodes 980 – 1000), GORGO optimized its strategy, reducing threads by 83.16% and requests per thread by 27.12%, shifting to 20 threads and 200 requests. Concurrently, fuzzing byte size increased by 23.07% to an average of 2700 bytes. Notably, this strategic optimization was not predetermined by GORGO's training parameters. The significant reduction in resources suggests autonomous decision-making. GORGO's shift toward more efficient resource use (i.e., decreasing threads and requests while increasing fuzzing byte size) reflects its own initiative beyond initial programming. This self-driven adaptation underscores GORGO's ability to independently refine its attack strategy.

Figure 21 shows GORGO's fuzzing byte decisions during B5G AMF training. The preferred fuzz data category was Format Characters, specifically Unicode range `0x20D0`–`0x20FF`. Notably, `Format Characters` were also preferred during AMF attack

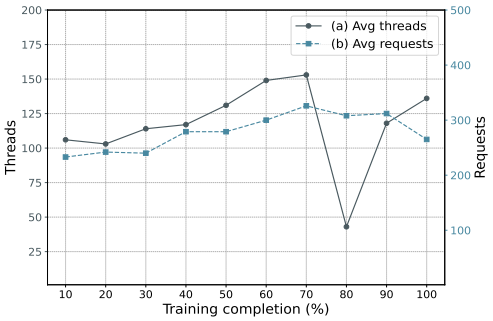

Fig. 19: Thread and request number preference on B5G AMF training

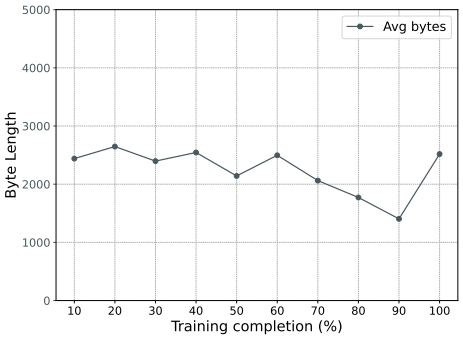

Fig. 20: Packet length preference on B5G AMF training

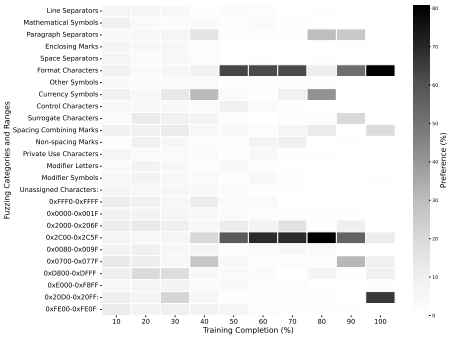

Fig. 21: Fuzzing byte categories and ranges preference during training on B5G AMF

Table 7: Optimal attack strategy on B5G AMF

Strategy	**Optimal selection**
Protocol	SCTP
Port	38412
Threads	20
Requests	200
Interval time (s)	0
Session management	Keep Open
Packet length	2700
Seed value	85
Characters	Format Characters
Unicode range	0x20D0-0x20FF

on port 38412 (SCTP). GORGO's optimal attack strategy is outlined in Table 7 illustrates the strategy of maintaining continuous active sessions by eliminating delays between requests, thus ensuring uninterrupted SCTP sessions. Specifically, the strategy involves spawning 20 threads, each issuing 200 requests, with a cumulative fuzzing payload size of 2700 bytes and a seed value of 85. The optimal fuzzing approach for the B5G AMF service leverages `Format Characters` and Unicode characters within the range `0x20D0-0x20FF`.

6 Comparative Analysis

The comparative analysis of DoS attack strategies across 5G and B5G network components reveals key differences in protocol vulnerabilities, resource allocation, and optimal attack parameters. Table 8 summarizes the optimal strategies identified by GORGO model for attacking various network functions, including 5G SMF, UPF, AMF and the B5G AMF. Notably, the SMF component showed high susceptibility to

Table 8: Comparison of DoS strategies on different services

Strategy	5G-SMF	5G-UPF	5G-AMF	B5G-AMF
Protocol	HTTP	UDP	TCP	SCTP
Port	8080	2152	5868	38412
Threads	60	10	90	20
Requests	240	300	340	200
Interval Time (s)	2	2	0	0
Session management	Close	Close	Keep Open	Keep Open
Packet Length	400	1300	2600	2700
Seed value	22	10	48	85
Characters	Currency Symbols	Paragraph Seperators	Enclosing	Format Characters
Unicode Range	0x0700-0x077F	0x0700-0x077F	0xD800-0xDFFF	0x20D0-0x20FF
Total DoS attacks	882	959	1000	1000

HTTP-based attacks targeting port 8080, employing 60 threads and 240 requests per thread with a moderate packet size of 400 bytes. Conversely, the UPF demonstrated vulnerability primarily to UDP attacks at port 2152, characterized by fewer threads (10) but a higher number of requests per thread (300) and significantly larger packet sizes (1300 bytes). The 5G AMF exhibited vulnerability through diverse strategies, utilizing both TCP and SCTP protocols across multiple ports with a significant number of threads (90) and requests (340), alongside the strategic choice of `Enclosing characters` from the Unicode range `0xD800-0xDFFF`. In contrast, the B5G AMF vulnerability was exploited using continuous SCTP sessions on a single port (38412), maintaining fewer threads (20) but ensuring persistent requests with zero interval time and larger packet sizes (2700 bytes), specifically using `Format Characters` from the Unicode range `0x20D0-0x20FF`. These details emphasize how the optimal attack strategies leverage specific Unicode character ranges and categories, highlighting the nuanced exploitation strategies utilized by the GORGO model.

More importantly, the growth in the total number of successful DoS attacks across 5G and B5G components, as shown in Table 8, is primarily attributed to the improved performance of the pretrained GORGO model compared to its initial, untrained state. Previous evaluations demonstrate significant training efficiency improvements—75.80% for UPF and up to 81.49% for AMF components—due to the reuse of pre-trained weights from earlier SMF attacks. This pretrained knowledge enabled GORGO to rapidly adapt and identify vulnerabilities more effectively, significantly reducing training time and enhancing attack success rates. The B5G AMF component, despite its more sophisticated infrastructure, was compromised with maximal efficiency by leveraging persistent open SCTP sessions and optimized packet parameters, clearly illustrating the strategic advantage gained through previous training. This analysis underscores the critical role that pretrained AI models play in both optimizing attack effectiveness and increasing the complexity of defense mechanisms required for emerging network technologies.

Figure 22 illustrates GORGO's CPU consumption profile, while Figure 23 presents RAM usage during penetration testing against each of the 5G+ core network components. Across all targets, GORGO follows a two-phase strategy: an initial CPU-intensive port scan to enumerate exposed services, followed by a tailored exploitation

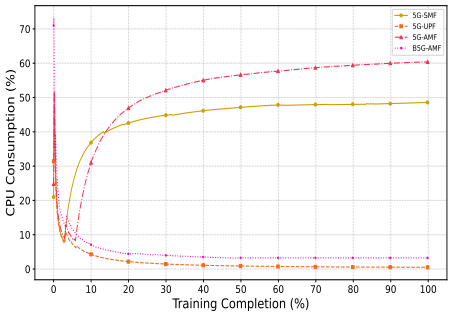

Fig. 22: GORGO's CPU consumption across 5G+ core components

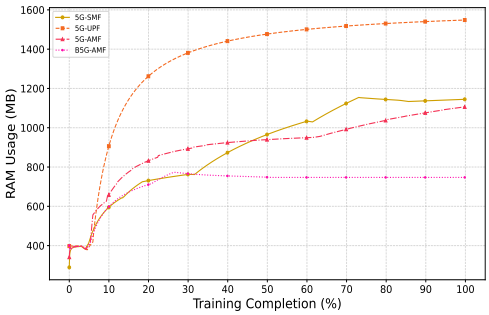

Fig. 23: GORGO's RAM usage across 5G+ core components

phase that induces component-specific resource strain. This process repeats iteratively as GORGO refines its strategies during training.

When evaluating the 5G SMF, GORGO initiates its scan on TCP ports 3868 and 5868 as well as the HTTP interface on port 8080. This scanning phase drives CPU consumption into the 30–50% range, as seen in the SMF curve in Figure 22, while RAM remains relatively stable below 400 MB (Figure 23). Once endpoint discovery completes and GORGO transitions into sustained session-level probing, CPU usage drops into the single-digit percentages but then climbs logarithmically until it peaks at around 50% utilization. Meanwhile, RAM usage begins a gradual and consistent rise from a 290 MB baseline to over 1.1 GB by the end of training. This steady increase reflects state accumulation due to the creation and teardown of hundreds of concurrent TCP and HTTP sessions, exploiting the SMF's stateful behavior.

In the case of the 5G UPF, a similar early CPU surge appears as GORGO scans for open UDP ports, causing utilization to reach around 30%. At this stage, RAM holds steady near 400 MB, since GORGO is not yet triggering UDP attacks. Once the optimal attack parameters are learned, the UPF's CPU footprint collapses to around 1%, reflecting the stateless and throughput-optimized evaluation tactic. However, RAM usage climbs sharply, peaking near 1.6 GB as GORGO overwhelms the UPF's packet buffers by injecting fuzzing and oversized traffic data. The disparity between low CPU load and high RAM pressure underscores the UPF's vulnerability to memory exhaustion despite minimal processing overhead.

The 5G AMF, which manages control-plane signaling and mobility procedures, presents a more balanced resource signature. GORGO's initial probe covers SCTP ports 36412 and 38412, along with TCP ports 3868 and 5868, lifting CPU utilization into the 20–25 percent range. Memory remains nearly flat during this phase. As targeted SCTP and TCP floods intensify, CPU demand rises progressively from 25% to over 60%. Concurrently, RAM usage increases from roughly 340 MB to about 1.1 GB. This behavior stems from the accumulation of subscriber and handover contexts, reflecting the AMF's sensitivity to connection overloads and its more complex signaling state machine.

Table 9: Comparison on state-of-the-art attacks on 5G and beyond networks

SOTA Methods	5G	B5G	AI Enabled	Fuzzing	Adaptive Strategy	Session Mgmt.	TCP	UDP	SCTP	HTTP	Other Prot.
PFCP attack [76]	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗
5Greplay [77]	✓	✗	✗	✓	✗	✗	✓	✓	✓	✓	✗
SliceSecure [18]	✓	✗	✗	✗	✗	✓	✓	✓	✗	✗	✗
R. Ettiane [78]	✓	✗	✗	✗	✗	✓	✗	✗	✗	✗	✓
H. Djuitcheu [79]	✓	✗	✗	✗	✗	✓	✗	✓	✗	✗	✗
Y. Shi [80]	✓	✗	✓	✗	✗	✓	✗	✗	✗	✗	✓
C. Chang [81]	✓	✓	✗	✗	✗	✓	✗	✗	✗	✓	✗
GORGO	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗

By contrast, the B5G AMF variant demonstrates a more efficient and hardened response profile. The port-scanning phase triggers a short-lived CPU spike above 70%, particularly when probing SCTP port 38412. RAM usage during this phase stays under 400 MB. Once GORGO identifies a viable exploitation strategy, CPU consumption rapidly drops into the low single-digit percentages and remains there for the duration of training. Meanwhile, memory usage stabilizes around 745 MB after an initial climb. This stabilization suggests a conscious shift toward reduced per-session state management, informed by previous experience.

The comparative analysis summarized in Table 9 highlights the distinctive capabilities and enhanced performance of GORGO compared to current state-of-the-art evaluation methods targeting 5G+ network infrastructures. Unlike existing methods such as PFCP attack [76], 5Greplay [77] and SliceSecure [18], GORGO uniquely targets both 5G and B5G infrastructures comprehensively. It demonstrates versatility by effectively exploiting multiple protocols including TCP, UDP, SCTP and HTTP, which significantly broadens its operational scope compared to other methods restricted to fewer protocols or specific vulnerabilities. Additionally, GORGO employs fuzzing methodologies complemented by an integrated Reinforcement Learning agent, session management and adaptive strategy adjustments. This comprehensive combination, absent in other comparative solutions, positions GORGO as notably superior in executing dynamic and complex attacks.

GORGO's sophisticated adaptive capabilities are underscored by its autonomous optimization of attack parameters, as evidenced through detailed evaluations across SMF, UPF and AMF components in both 5G and B5G environments. Its adaptive strategy includes dynamic adjustments in threads, request intervals, and fuzzing payload characteristics based on real-time feedback, distinguishing it from prior solutions such as those proposed by Ettiane [78], Djuitcheu [79] and Chang [81] which lack such real-time adaptability. Furthermore, the strategic use of pre-trained agents from earlier network interactions has proven essential in reducing learning times significantly and enhancing overall attack effectiveness. GORGO's ability to maintain persistent sessions and refine attack vectors autonomously illustrates its advanced self-learning capabilities, significantly raising the bar for defense mechanisms in emerging 5G+ network technologies.

Table 10: Comparison of Adversarial and Vulnerability Testing Approaches

Aspect	**Shi et al. [80]**	**GORGO**
Primary Focus	Adversarial attack via resource starvation	Penetration testing & vulnerability assessment
Attack Strategy	Fake request flooding	Protocol-level DoS & sophisticated fuzzing attacks
Reinforcement Learning Type	Q-Learning (traditional)	Multi-agent Deep Q-Learning
Goal	Minimize legitimate requests' QoE (resource allocation)	Quantify and test resilience against various network disruptions
Environment	Virtual Resource Blocks in RAN slicing	Real-world protocols (TCP, UDP, SCTP, HTTP) in 5G/B5G infrastructure
Testing Approach	Simulations with strategic fake requests	Real 5G and B5G environments using advanced packet generation
Scope	Theoretical adversarial scenario exploration	Practical real-world vulnerability testing tool

In recent literature, Reinforcement Learning approaches have been explored for evaluating and mitigating vulnerabilities within 5G network infrastructures. To the best of our knowledge, the only existing work that applies Reinforcement Learning in this context is the study by Shi et al. [80]. Their approach introduced an adversarial methodology leveraging traditional Q-learning to orchestrate resource starvation attacks in 5G RAN slicing. This strategy involved generating fake slicing requests to exploit resource allocation mechanisms, focusing narrowly on simulated RAN environments and avoiding deeper interactions with transport or session-level protocols.

However, as shown in Table 10, the scope and depth of Shi's approach are considerably limited when compared to GORGO's capabilities. The methodology introduced by our work, significantly expands the scope and effectiveness of vulnerability assessment through the application of DQN, thus surpassing the limitations of conventional Q-learning as employed by Shi. GORGO employs multiple collaborating DQN agents capable of autonomously constructing sophisticated and adaptive DoS attacks, simultaneously targeting multiple critical network protocols including TCP, UDP, SCTP and HTTP. Additionally, GORGO integrates adaptive strategy techniques and advanced protocol-level fuzzing methods to identify and exploit potential vulnerabilities more effectively, reflecting realistic and evolving threat landscapes, while Shi's approach remains confined to simulated scenarios based solely on RAN slicing requests.

Furthermore, GORGO provides detailed real-time insights into the impact of DoS attacks by precisely quantifying QoS degradation through critical metrics such as latency, throughput and packet loss. This method delivers more comprehensive and practically relevant evaluations, offering valuable insights into specific network vulnerabilities and their operational consequences. Shi's methodology, conversely, measures

the effectiveness of attacks indirectly by evaluating resource depletion through simulated reinforcement learning rewards, without directly assessing specific QoS metrics critical for real-world deployments.

The practical validation of GORGO in real-world 5G and B5G testbed environments underscores its practical applicability and effectiveness in identifying vulnerabilities that traditional or simulation-based assessments might miss. Shi's methodology remains predominantly theoretical, with validation relying entirely on simulation outcomes, thus lacking real-world network complexity and unpredictability.

Therefore, the multi-protocol capability, adaptive strategy techniques, advanced fuzzing capabilities, real-world applicability, and sophisticated QoS evaluation clearly distinguish GORGO as a superior methodology for evaluating 5G and B5G network resilience against advanced AI-driven DoS threats.

7 Discussion

This section synthesizes the key empirical findings, evaluates their methodological and practical implications, and acknowledges the study's limitations.

7.1 Lessons Learned

GORGO demonstrated that multi-agent Deep Q-Learning approach can effectively uncover vulnerabilities in 5G+ core network components and evaluate their impact on adversarial attacks. First, the pronounced shift from brute-force resource consumption toward precision attacks evidenced by GORGO's autonomous reduction in thread and request counts while substantially increasing payload size, underscores the importance of packet-level sophistication over mere volumetric approaches. Second, the model's convergent preference for specific Unicode character categories (e.g., `Currency Symbols` or `Format Characters`) and byte ranges highlights that even subtle variations in packet content can dramatically affect protocol handling, suggesting defenders must consider deep-content inspection rather than coarse protocol filters. Third, the seamless transfer of pre-trained weights from 5G SMF to UPF and AMF components demonstrated the value of cross-component learning: once GORGO internalized effective attack patterns, it achieved near-perfect success rates on subsequent services with minimal additional training time. Fourth, the uniformly high vulnerability of services, from HTTP port 8080 in SMF to SCTP port 38412 in B5G AMF, reinforces that AI-enabled DoS strategies pose a universal threat across both control and user-plane functions, compelling operators to adopt proactive, adaptive defense mechanisms that evolve in parallel with attacker intelligence. Finally, GORGO's ability to both conduct fine-grained protocol-level fuzzing and provide real-time QoS degradation metrics (latency, throughput, packet loss) illustrates the dual role such tool must play: not only as penetration testing tool but also as generator of training data for next-generation IDS, thereby closing the feedback loop between attack discovery and detection model refinement.

7.2 Limitations and Future Work

Although GORGO effectively learns high-impact DoS strategies, our current evaluation does not measure attack stealth against intrusion detection or mitigation solutions. As a next step, we plan to test GORGO's generated traffic against representative detection systems and open-source anomaly detection tools. This will help to quantify evasion rates and identify features that enable bypassing signature-, behavior-, and machine learning–based defenses. We will further extend this evaluation by integrating state-of-the-art Deep Learning-driven intrusion detection systems to assess GORGO's ability to evade learning-based defenses. In addition, we will evaluate GORGO against more general network infrastructures and Internet of Things devices to assess its resilience and vulnerability to adversarial attacks in diverse deployment scenarios. The study also acknowledges that the transfer sequence may produce comparable results; therefore, a systematic comparison of alternative orderings will be incorporated into future work, examining success rates, sample efficiency and stability across all possible permutations. Integrating these analyses will complete the attack lifecycle assessment and guide the development of more robust countermeasures.

8 Related Work

While 5G+ networks enhance connectivity and enable advanced applications, they simultaneously introduce a range of security vulnerabilities, particularly in the context of network slicing. A prominent concern is the susceptibility of network functions to DoS and Distributed Denial of Service (DDoS) attacks, which can compromise the integrity and performance of isolated network slices.

Shi and Sagduyu [80] present a Reinforcement Learning (RL)-based flooding attack targeting simulated 5G RAN slicing. The adversary strategically generates fake slicing requests to deplete virtual resources and reduce service quality for legitimate users. Their study highlights the susceptibility of RL-based resource allocation mechanisms to adversarial exploitation and demonstrates how such attacks outperform traditional methods by evading detection through techniques like randomized weight masking.

Amponis et al. [76] delve into vulnerabilities within the 5G network's Packet Forwarding Control Protocol (PFCP), highlighting the risk of DoS attacks on Unmanned Aerial Vehicle communications. Their research underscores how unauthorized control packets can silently disrupt 5G tunnels, particularly within the 5GC and NG-RAN. Their work not only reveals security flaws in control plane communications but also contributes to the development of datasets for AI-based intrusion and anomaly detection systems in 5G environments.

Expanding on attack modeling, Salazar et al. [77] introduce 5Greplay, an ontology rooted in mutation techniques designed to simulate cybersecurity attacks within 5G networks. Their framework enables the generation of realistic network traffic using mutation operators, facilitating both known and unknown attack scenarios. This approach lays the groundwork for robust fuzz testing and enhances the fidelity of security evaluation tools in 5G.

Focusing on the direct impact of DoS threats on network performance, Khan et al. [18] simulate attacks in a 5G network slice environment to evaluate how compromised UE can overload virtualized functions such as the UPF. Their findings reveal significant degradation in bandwidth and latency, emphasizing the need for targeted defense mechanisms. To counter these issues, they propose a deep-learning-based detection system capable of accurately identifying and mitigating such attacks. The authors further contribute by releasing datasets specifically tailored for DoS/DDoS attack scenarios in 5G slice environments.

Ettiane et al. [78] further examine vulnerabilities within the 5G control plane, focusing on attacks exploiting new features of the Radio Resource Control (RRC) protocol. They identify two impactful DoS attacks: one that manipulates RRC state transitions, causing excessive signaling overload, and another novel attack utilizing massive on-demand System Information Block (SIB) requests to flood the random access procedure. Simulations reveal these attacks significantly degrade network performance, highlighting the need for effective defense mechanisms such as parameter randomization to mitigate these threats.

Complementing simulation-based work with real-world experimentation, Djuitcheu et al. [79] conduct an empirical assessment of DDoS impacts using a 5G standalone testbed built with the Amarisoft Callbox. By deploying multiple UEs, including malicious nodes, they assess performance metrics like latency, jitter, bitrate, and packet loss under attack conditions. Their results, based on tools such as Hping3 [15] and LOIC [82], demonstrate substantial performance degradation, offering concrete evidence of 5G's vulnerability to resource exhaustion attacks beyond theoretical models.

Building upon previous studies, Chang et al. [81] further explore vulnerabilities in 6G roaming scenarios, focusing specifically on DoS attacks targeting HTTP/2 within the 5G Authentication and Key Agreement (5G-AKA). Using Open5GS [75], they identify the network's inability to detect malicious registration requests and uncover a critical flaw in the 5G-AKA, which attackers exploit through fraudulent authentication responses. Their experimental findings underscore that roaming significantly amplifies the impact of DoS attacks compared to non-roaming conditions, highlighting critical risks to network availability.

To address these emerging threats, recent research has explored Reinforcement Learning-based defense mechanisms. For instance, Slice Isolation-based Reinforcement Learning (SIRL) [83] introduces an actor-critic model for slice isolation, which not only mitigates DoS effects but also optimizes the mapping of slice requests to maintain service quality. Looking ahead to 6G, the integration of AI-driven functions raises additional concerns. Hoang et al. [84] discuss adversarial threats targeting key 6G components, including open RAN and federated learning. They propose countermeasures such as adversarial training and anomaly detection to enhance the robustness of next-generation mobile networks.

9 Conclusion

This paper presents GORGO, a multi-agent Reinforcement Learning model that significantly advances the vulnerability analysis for 5G+ networks. By utilizing DQN agents, GORGO autonomously identifies weaknesses in real-world infrastructures, providing insights into the exploitation phase. Its capability to model complex attacks, perform protocol-level fuzzing and evaluate both 5G and B5G environments highlights its versatility as a penetration testing tool. The evaluation found that the SMF component in 5G infrastructure is vulnerable via HTTP port 8080, while the UPF component exhibited vulnerabilities on UDP port 2152. Additionally, the 5G AMF was found susceptible to attacks on TCP ports 3858 and 5868, as well as SCTP ports 36412 and 38412. Regarding the B5G infrastructure, vulnerability was identified on SCTP port 38412. These findings underscore the importance of proactive and adaptive defense mechanisms to secure modern network infrastructures against evolving AI-enabled threats. GORGO's contribution to enhancing resilience and improving intrusion detection systems marks a crucial step in safeguarding next-generation communication technologies.

Acknowledgments

CERTH was responsible for the conceptualization of the research idea, design, implementation, and evaluation of GORGO, developed under the Horizon 2020 (SANCUS – GA No. 952672) and Horizon Europe (ARROW – in the context of 6G-SANDBOX GA No. 101096328) research and innovation programs. University of Piraeus was responsible for the entire writing, presentation, and review of the manuscript. The authors have agreed that GORGO's intellectual property rights reside with CERTH under the funding of the SANCUS and ARROW projects.

References

[1] 3GPP: 5G System Overview. https://www.3gpp.org/technologies/5g-system-overview. (Accessed: March 2025)

[2] Manda, J.K.: 5g-enabled smart cities: Security and privacy considerations. Innovative Engineering Sciences Journal **4**(1) (2024)

[3] Suraci, C., Araniti, G., Abrardo, A., Bianchi, G., Iera, A.: A stakeholder-oriented security analysis in virtualized 5g cellular networks. Computer Networks **184**, 107604 (2021)

[4] Yamin, M.M., Ullah, M., Ullah, H., Katt, B.: Weaponized ai for cyber attacks. Journal of Information Security and Applications **57**, 102722 (2021)

[5] Charfeddine, M., Kammoun, H.M., Hamdaoui, B., Guizani, M.: Chatgpt's security risks and benefits: offensive and defensive use-cases, mitigation measures, and future implications. IEEE Access (2024)

[6] Zolotukhin, M., Zhang, D., Hämäläinen, T., Miraghaei, P.: On attacking future 5g networks with adversarial examples: Survey. Network **3**(1), 39–90 (2022)

[7] Lilhore, U.K., Dalal, S., Simaiya, S.: A cognitive security framework for detecting intrusions in iot and 5g utilizing deep learning. Computers & Security **136**, 103560 (2024)

[8] Scalingi, A., D'Oro, S., Restuccia, F., Melodia, T., Giustiniano, D.: Det-ran: Data-driven cross-layer real-time attack detection in 5g open rans. In: IEEE INFOCOM 2024-IEEE Conference on Computer Communications, pp. 41–50 (2024). IEEE

[9] Rafique, W., Barai, J., Fapojuwo, A.O., Krishnamurthy, D.: A survey on beyond 5g network slicing for smart cities applications. IEEE Communications Surveys & Tutorials (2024)

[10] Geetha, A., Kumari, P.: Network virtualization. In: Towards Wireless Heterogeneity in 6G Networks, pp. 226–242. CRC Press, FL, Boca Raton, U.S. (2024)

[11] Rustam, F., Ranaweera, P., Jurcut, A.D.: Ai on the defensive and offensive: Securing multi-environment networks from ai agents. In: ICC 2024-IEEE International Conference on Communications, pp. 4287–4292 (2024). IEEE

[12] Roach, K.: Llms for malware offense and defense

[13] WOLFGPT Project: Website. https://github.com/ianwolf99/WOLFGPT. (Accessed: March 2025)

[14] Nmap: Website. https://nmap.org/. (Accessed: March 2025)

[15] Hping3: Website. https://www.kali.org/tools/hping3/. (Accessed: March 2025)

[16] lksctp-tools: Website. https://www.commandlinux.com/man-page/man1/sctp_test.1.html. (Accessed: March 2025)

[17] Salazar, Z., Nguyen, H.N., Mallouli, W., Cavalli, A.R., Oca, E.: 5greplay: A 5g network traffic fuzzer-application to attack injection. In: Proceedings of the 16th International Conference on Availability, Reliability and Security, pp. 1–8 (2021)

[18] Khan, M.S., Farzaneh, B., Shahriar, N., Saha, N., Boutaba, R.: Slicesecure: Impact and detection of dos/ddos attacks on 5g network slices. In: 2022 IEEE Future Networks World Forum (FNWF), pp. 639–642 (2022). IEEE

[19] Ghahramani, Z.: Unsupervised learning. In: Summer School on Machine Learning, pp. 72–112. Springer, Berlin, Germany (2003)

[20] Deliu, N., Williams, J.J., Chakraborty, B.: Reinforcement learning in modern biostatistics: constructing optimal adaptive interventions. International Statistical

Review (2024)

[21] Ebadinezhad, S., Conteh, A., Ali, K.A., Hussein, I.K.: Reinforcement learning for dynamic traffic routing and optimization. In: 2024 International Conference on Inventive Computation Technologies (ICICT), pp. 346–352 (2024). IEEE

[22] Sciancalepore, V., Costa-Perez, X., Banchs, A.: Rl-nsb: Reinforcement learning-based 5g network slice broker. IEEE/ACM transactions on networking **27**(4), 1543–1557 (2019)

[23] Wang, J., Li, J., Liu, J.: Digital twin-assisted flexible slice admission control for 5g core network: A deep reinforcement learning approach. Future Generation Computer Systems **153**, 467–476 (2024)

[24] Puterman, M.L.: Markov decision processes. Handbooks in operations research and management science **2**, 331–434 (1990)

[25] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge, Massachusetts, U.S. (2018)

[26] Strens, M.: A bayesian framework for reinforcement learning. In: ICML, vol. 2000, pp. 943–950 (2000)

[27] Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Hoboken, New Jersey, U.S. (2014)

[28] Bellman, R.: Dynamic programming and stochastic control processes. Information and control **1**(3), 228–239 (1958)

[29] Kanervisto, A., Scheller, C., Hautamäki, V.: Action space shaping in deep reinforcement learning. In: 2020 IEEE Conference on Games (CoG), pp. 479–486 (2020). IEEE

[30] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[31] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., *et al.*: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

[32] Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30 (2016)

[33] Watkins, C.J., Dayan, P.: Q-learning. Machine learning **8**, 279–292 (1992)

[34] Clifton, J., Laber, E.: Q-learning: Theory and applications. Annual Review of Statistics and Its Application **7**(1), 279–301 (2020)

[35] Bellman, R.: The theory of dynamic programming. Bulletin of the American Mathematical Society **60**(6), 503–515 (1954)

[36] Tran, H.V.: Hamilton–Jacobi Equations: Theory and Applications vol. 213. American Mathematical Soc., Providence, Rhode Island (2021)

[37] Ha, V.T., Tu, T.N., Dung, N.T., Mien, T.L., Thuy, C.T.: Deep q-network (dqn) approach for automatic vehicles applied in the intelligent transportation system (its). In: 2023 International Conference on System Science and Engineering (ICSSE), pp. 527–532 (2023). IEEE

[38] Gupta, A., Jha, R.K.: A survey of 5g network: Architecture and emerging technologies. IEEE access **3**, 1206–1232 (2015)

[39] Wen, M., Li, Q., Kim, K.J., López-Pérez, D., Dobre, O.A., Poor, H.V., Popovski, P., Tsiftsis, T.A.: Private 5g networks: Concepts, architectures, and research landscape. IEEE Journal of Selected Topics in Signal Processing **16**(1), 7–25 (2021)

[40] Dahlman, E., Mildh, G., Parkvall, S., Peisa, J., Sachs, J., Selén, Y.: 5g radio access. Ericsson review **6**(1) (2014)

[41] Lee, W., Na, T., Kim, J.: How to create a network slice?-a 5g core network perspective. In: 2019 21st International Conference on Advanced Communication Technology (ICACT), pp. 616–619 (2019). IEEE

[42] Choi, Y.-i., Park, N.: Slice architecture for 5g core network. In: 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 571–575 (2017). IEEE

[43] Gures, E., Shayea, I., Alhammadi, A., Ergen, M., Mohamad, H.: A comprehensive survey on mobility management in 5g heterogeneous networks: Architectures, challenges and solutions. IEEE Access **8**, 195883–195913 (2020)

[44] Fondo-Ferreiro, P., Gil-Castiñeira, F., González-Castaño, F.J., Candal-Ventureira, D.: A software-defined networking solution for interconnecting network functions in service-based architectures. IEEE Access **10**, 19905–19916 (2022)

[45] Tang, Q., Ermis, O., Nguyen, C.D., De Oliveira, A., Hirtzig, A.: A systematic analysis of 5g networks with a focus on 5g core security. IEEE Access **10**, 18298–18319 (2022)

[46] Gökarslan, K., Sandal, Y.S., Tugcu, T.: Towards a urllc-aware programmable data path with p4 for industrial 5g networks. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6 (2021). IEEE

[47] Phung, K.-H., Dung, L.V., Thai, Q.T., Ko, N.: User plane management function: A solution for automatic deployment of upf on cloud-native 5g core network architecture. In: 2023 14th International Conference on Information and Communication Technology Convergence (ICTC), pp. 773–778 (2023). IEEE

[48] Rivera, J.J.D., Khan, T.A., Mehmood, A., Song, W.-C.: Network slice selection function for data plane slicing in a mobile network. In: 2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), pp. 1–4 (2019). IEEE

[49] Rommer, S., Hedman, P., Olsson, M., Frid, L., Sultana, S., Mulligan, C.: 5G Core Networks: Powering Digitalization. Academic Press, Cambridge, Massachusetts, U.S. (2019)

[50] Lanov, D.: Shortest path assisted user plane function selection on 5g session management function (2022)

[51] Arteaga, C.H.T., Ordoñez, A., Rendon, O.M.C.: Scalability and performance analysis in 5g core network slicing. Ieee Access **8**, 142086–142100 (2020)

[52] Vittal, S., *et al.*: Self optimizing network slicing in 5g for slice isolation and high availability. In: 2021 17th International Conference on Network and Service Management (CNSM), pp. 125–131 (2021). IEEE

[53] Robitzsch, S., Sarathchandra, C., Starsinic, M., De Foy, X.: Formation and assertion of data unit groups in 3gpp networks with tsn and pdu set support. In: 2024 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6 (2024). IEEE

[54] Du, K., Wen, X., Wang, L., Nguyen, T.-T.: A cloud-native based access and mobility management function implementation in 5g core. In: 2020 IEEE 6th International Conference on Computer and Communications (ICCC), pp. 1251–1256 (2020). IEEE

[55] Alawe, I., Hadjadj-Aoul, Y., Ksentini, A., Bertin, P., Darche, D.: On the scalability of 5g core network: The amf case. In: 2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC), pp. 1–6 (2018). IEEE

[56] ETSI TS: 5G Security Assurance Specification (SCAS) Access and Mobility management Function (AMF) (3GPP TS 33.512 version 17.3.0 Release 17). https://www.etsi.org/deliver/etsi_ts/133500_133599/133512/17.03.00_60/ts_133512v170300p.pdf. (Accessed: March 2025)

[57] Ha, J., Choi, Y.-I.: Support of a multi-access session in 5g mobile network. In: 2019 25th Asia-Pacific Conference on Communications (APCC), pp. 378–383 (2019). IEEE

[58] Chen, J., Wang, L., Liu, Y., Lu, Z.: Signaling priority reconstruction for micro-serviced amf in 6g. In: 2024 IEEE Wireless Communications and Networking Conference (WCNC), pp. 01–06 (2024). IEEE

[59] Stewart, R., Metz, C.: Sctp: new transport protocol for tcp/ip. IEEE Internet Computing **5**(6), 64–69 (2001)

[60] Jain, A., Lopez-Aguilera, E., Demirkol, I.: Improved handover signaling for 5g networks. In: 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pp. 164–170 (2018). IEEE

[61] Qian, S., Chen, F., Ning, H., Lin, T., Yuanyuan, W.: Computing and storage resources allocation of upf based on isolation in private 5g networks. In: 2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring), pp. 1–6 (2022). IEEE

[62] Leyva-Pupo, I., Cervelló-Pastor, C., Anagnostopoulos, C., Pezaros, D.P.: Dynamic scheduling and optimal reconfiguration of upf placement in 5g networks. In: Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 103–111 (2020)

[63] Collins, J.: Gprs tunneling protocol (gtp). In: Encyclopedia of Cryptography, Security and Privacy, pp. 1–3. Springer, Berlin, Germany (2022)

[64] Amaral, T.A.N., Rosa, R.V., Moura, D.F.C., Rothenberg, C.E.: An in-kernel solution based on xdp for 5g upf: Design, prototype and performance evaluation. In: 2021 17th International Conference on Network and Service Management (CNSM), pp. 146–152 (2021). IEEE

[65] Zeydan, E., Turk, Y.: User plane acceleration service for next-generation cellular networks. Telecommunication Systems **84**(4), 469–485 (2023)

[66] Mazhar, T., Malik, M.A., Mohsan, S.A.H., Li, Y., Haq, I., Ghorashi, S., Karim, F.K., Mostafa, S.M.: Quality of service (qos) performance analysis in a traffic engineering model for next-generation wireless sensor networks. Symmetry **15**(2), 513 (2023)

[67] Polyakov, N., Platonova, A.: Assessing latency of packet delivery in the 5g 3gpp integrated access and backhaul architecture with half-duplex constraints. Future Internet **14**(11), 345 (2022)

[68] Jain, S., Jain, R., Sharma, J., Sharma, A.: Improving performance and quality of service in 5g technology. In: Mobile Radio Communications and 5G Networks: Proceedings of Second MRCN 2021, pp. 439–449. Springer, Berlin, Germany (2022)

[69] Irazabal, M., Lopez-Aguilera, E., Demirkol, I.: Active queue management as quality of service enabler for 5g networks. In: 2019 European Conference on Networks and Communications (EuCNC), pp. 421–426 (2019). IEEE

[70] Kalafatidis, S., Agrafiotis, G., Giapantzis, K., Lalas, A., Votis, K.: Experiments with digital security processes over sdn-based cloud-native 5g core networks. In: 2024 27th Conference on Innovation in Clouds, Internet and Networks (ICIN), pp. 97–99 (2024). IEEE

[71] 6G SANDBOX Project: 6G SANDBOX Project Website. https://6g-sandbox.eu/pilot-6g-sites/malaga/. (Accessed: March 2025)

[72] Agrafiotis, G., Kalafatidis, S., Giapantzis, K., Lalas, A., Votis, K.: Advancing cybersecurity with ai: A multimodal fusion approach for intrusion detection systems. In: 2024 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), pp. 51–56 (2024). IEEE

[73] free5GC: Website. https://free5gc.org/. (Accessed: March 2025)

[74] Kalafatidis, S., Mamatas, L.: Microservices-adaptive software-defined load balancing for 5g and beyond ecosystems. IEEE network **36**(6), 46–53 (2022)

[75] Open5GS: Website. https://github.com/open5gs/open5gs. (Accessed: March 2025)

[76] Amponis, G., Radoglou-Grammatikis, P., Lagkas, T., Mallouli, W., Cavalli, A., Klonidis, D., Markakis, E., Sarigiannidis, P.: Threatening the 5g core via pfcp dos attacks: the case of blocking uav communications. EURASIP Journal on Wireless Communications and Networking **2022**(1), 124 (2022)

[77] Salazar, Z., Zaïdi, F., Nguyen, H.-N., Mallouli, W., Cavalli, A.R., De Oca, E.M.: A network traffic mutation based ontology, and its application to 5g networks. IEEE Access **11**, 43925–43944 (2023)

[78] Ettiane, R., Chaoub, A., Elkouch, R.: Toward securing the control plane of 5g mobile networks against dos threats: Attack scenarios and promising solutions. Journal of Information Security and Applications **61**, 102943 (2021)

[79] Djuitcheu, H., Shah, T., Tammen, M., Schotten, H.D.: Ddos impact assessment on 5g system performance. In: 2023 IEEE Future Networks World Forum (FNWF), pp. 1–6 (2023). IEEE

[80] Shi, Y., Sagduyu, Y.E.: Adversarial machine learning for flooding attacks on 5g radio access network slicing. In: 2021 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6 (2021). IEEE

[81] Chang, C.-H., Chang, R.-L., Chen, H.-Y., Lin, T.-N.: 6g security: The vulnerability of roaming technology via dos exploit of signaling control plane. In: 2024 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1413–1418 (2024). IEEE

[82] NewEraCracker: Website. https://github.com/NewEraCracker/LOIC. (Accessed: March 2025)

[83] Javadpour, A., Ja'fari, F., Taleb, T., Benzaïd, C.: Reinforcement learning-based slice isolation against ddos attacks in beyond 5g networks. IEEE Transactions on Network and Service Management **20**(3), 3930–3946 (2023)

[84] Hoang, V.-T., Ergu, Y.A., Nguyen, V.-L., Chang, R.-G.: Security risks and countermeasures of adversarial attacks on ai-driven applications in 6g networks: A survey. Journal of Network and Computer Applications, 104031 (2024)