Dynamic Energy Management in 5G and Beyond Wireless Networks Using Reinforcement Learning

Chrysostomos-Athanasios Katsigiannis
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: up1072490@upnet.gr

Konstantinos Tsachrelias
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: up1096511@upatras.gr

Vasileios Kokkinos
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: kokkinos@cti.gr

Apostolos Gkamas
Department of Chemistry
University of Ioannina
Ioannina, Greece
Email: gkamas@uoi.gr

Christos Bouras
Computer Engineering and Informatics Department
University of Patras
Patras, Greece
Email: bouras@upatras.gr

Philippos Pouyioutas
Computer Science Department
University of Nicosia
Nicosia, Cyprus
Email: pouyioutas.p@unic.ac.cy

***Abstract*- As the demand for high-speed, low-latency connectivity escalates, fifth generation (5G) and emerging sixth generation (6G) networks face significant challenges in managing energy consumption while maintaining performance standards. This paper investigates the application of Reinforcement Learning (RL) for dynamic energy management in these advanced wireless networks. We propose an RL based framework that enables Base Stations (BS) to adaptively adjust their operational states—such as active, idle, or sleep modes based on real-time network conditions and traffic demands. The objective is to minimize total network energy consumption without compromising Quality of Service (QoS) metrics such as latency and Service-Level Agreement (SLA) compliance. Through simulations, we demonstrate that our RL approach outperforms traditional static and heuristic methods in reducing energy consumption and enhancing network efficiency. The findings underscore the potential of RL in facilitating intelligent, energy-aware operations in next-generation wireless networks.**

Keywords- Reinforcement Learning, 5G Networks, 6G & Beyond Networks, Heterogeneous Networks, Energy Efficiency.

I. INTRODUCTION

The fifth generation (5G) of cellular networks has dramatically expanded the capacity and versatility of mobile communications, enabling applications such as autonomous driving, tactile internet services, and immersive extended reality. However, this expansion has also increased energy consumption in the Radio Access Network (RAN), raising concerns about the sustainability of large-scale 5G deployments [1]. As research progresses toward sixth generation (6G) systems that promise terabit-class data rates, pervasive intelligence, and integrated sensing, the challenge of reconciling ultra-high performance with strict energy budgets becomes even more critical [2].

Conventional energy-saving methods in wireless networks typically rely on fixed duty cycles or simple heuristic rules, such as predefined Base Station (BS) sleep schedules. Although these approaches can reduce power consumption under stable traffic conditions, they respond poorly to time-varying demand and channel fluctuations, which can either waste energy during low-load periods or degrade service quality when demand changes rapidly [3]. Their lack of adaptability is especially limited in dense and heterogeneous deployments, where traffic levels and QoS requirements vary significantly across cells and over time.

Reinforcement learning (RL) offers a more adaptive alternative by learning BS power-control decisions directly from repeated interaction with the network environment. In the proposed setting, the agent observes the current network condition and decides which BSs remain active and which enter sleep mode. By optimizing a reward that balances energy consumption and QoS, RL can support more efficient resource management than static control policies [4]. This dynamic, data-driven approach allows for more adaptive and efficient resource management compared to traditional methods.

Earlier studies based on Q-learning and Deep Q-Networks (DQNs) have shown that RL can support radio resource allocation, interference management, and energy-aware network control [5]. More generally, prior work on energy management in wireless networks spans fixed-rule sleep scheduling, heuristic traffic-aware switching, supervised-learning-based traffic prediction, and RL-based control. Static and heuristic approaches are lightweight and easy to implement, but they generally lack robustness under highly variable traffic conditions. Supervised-learning approaches can enhance traffic prediction, although they still require a separate control mechanism to translate forecasts into operational decisions. In contrast, RL-based methods are well suited to sequential decision-making in dynamic environments because they learn control policies directly from interaction with the system. Motivated by these observations, this paper develops an RL framework for energy management in 5G and beyond wireless networks. The framework formulates BS activation as a Markov Decision Process (MDP), incorporates energy consumption, latency-related QoS, and feasibility constraints into the control process, and evaluates the learned policy under realistic traffic and mobility patterns. Hence, the novelty of this work lies not in introducing RL to wireless energy management, but in formulating and evaluating a binary BS activation framework tailored to the considered 5G deployment scenario and benchmarked against static and heuristic baselines.

Compared with static and heuristic baselines, the proposed RL approach introduces additional computational overhead,

particularly during offline training. Static and threshold-based policies require only simple rule evaluation, whereas RL involves repeated parameter updates over a dataset of network transitions. However, this added cost is incurred mainly during the training phase. During runtime, policy execution is limited to a forward pass through the trained network followed by a feasibility-correction step, so the online decision process is substantially lighter than offline training. Therefore, the results reported in this paper focus on operational energy savings at the BS level. The computational energy cost associated with offline model training is acknowledged as an important practical consideration and is identified as a direction for future investigation. Building on this direction, this paper develops a unified RL framework for energy management in 5G and beyond wireless networks. The framework models BS activation as a Markov Decision Process (MDP), evaluates the learned policy under realistic traffic and mobility patterns, and compares it against static and heuristic baselines.

The remainder of the paper is organized as follows. Section II formulates the mathematical model underpinning the energy management problem. Section III presents the RL algorithm and analyzes its operational dynamics. Section IV describes the simulation environment, including network topology, traffic models, and key parameters. Section V evaluates performance across a range of scenarios, comparing the proposed RL policy against static and heuristic baselines. Finally, Section VI summarizes the main findings and outlines directions for future research.

II. Mathematical Model

The energy-management problem in the considered O-RAN is formulated as a MDP (S,A,T,r,γ). At each discrete decision epoch t, the controller observes a compact representation of the network state, selects an action that determines which cells remain active, and receives a scalar reward that reflects the trade-off between energy consumption and quality of service.

The paper considers deployment of N BSs, including one macro cell and several small cells. For each cell i∈ {1, ⋯, N}, the simulator maintains a vector of key performance metrics (KPMs), such as normalized load, latency proxy, energy efficiency, and channel quality, along with a binary indicator of the cell's operating mode. The per-cell feature vector at time t is denoted by:

$$xt(i) = [\ell t(i), \tau t(i), \eta t(i), ct(i), \chi t(i)] \qquad (1)$$

where $\ell_t^{(i)}$ represents the current cell load, $\tau_t^{(i)}$ a latency-related metric, $\eta_t^{(i)}$ the energy efficiency, $c_t^{(i)}$ a channel-quality indicator, and $\chi_t^{(i)} \in \{0,1\}$ a status flag with $\chi_t^{(i)} = 1$ for active and $\chi_t^{(i)} = 0$ for sleep mode. When $\chi_t^{(i)} = 1$ the BS operates in active mode if at least one UE is associated with it; otherwise, it remains in idle mode. All continuous features are clipped and normalized to $[0,1]$ for stable neural-network training. The global network state is then given by:

$$st = (xt(1), \dots, xt(N)) \in RN \times d \qquad (2)$$

where d is the dimension of the per-cell feature vector. This state captures the instantaneous operating condition of the RAN, including traffic load, service quality, and current energy footprint.

The controller action at time t is expressed as a binary vector:

$$at = (at(1), \dots, at(N)) \in \{0,1\}N \qquad (3)$$

where $a_t^{(i)} = 1$ instructs cell i to operate in active mode, and $a_t^{(i)} = 0$ requests a transition to or maintenance of sleep mode. To avoid coverage holes, the feasible action space is restricted by a minimum-active constraint,

$$Afeasible(st) = \{at \in \{0,1\}N : \sum_{i=1}^{N} at(i) \geq Nmin\} \quad (4)$$

where $N_{\min}$ is the minimum number of active cells enforced by the operator. In the experiments, we set $N_{\min} = 2$. Actions that do not satisfy this constraint are corrected by re-activating the highest-load cells until the minimum is met.

The environment transition kernel $T(s_{t+1} \mid s_t, a_t)$ is implemented by a lightweight Python simulator that emulates realistic 5G/6G traffic and interference dynamics. Given the current state and an action, the simulator updates user positions, associates users with cells according to distance and channel quality, computes inter-cell interference, and updates the per-cell KPMs and BS operating states. Time-of-day traffic profiles generate realistic load fluctuations with morning, midday, and evening peaks, while a night valley represents low-demand periods. Because the next state is computed solely from (s_t, a_t) and the internal randomization of the simulator, the Markov property holds by construction.

The reward function is designed to penalize high power draw while preserving latency and Service-Level Agreement (SLA) compliance. Let $P_i(a_t^{(i)})$ denote the power consumed by BS i under action $a_t^{(i)}$, taking values 400 W for an active macro cell, 120 W for an active small cell, 80 W for idle macro, and 15 W for sleep mode macro. Idle and sleep power levels are defined for macro base stations, which may operate in reduced-power modes under low traffic conditions, while small cells are modeled using a single active power level. The total power at time t is:

$$P_t^{\text{tot}} = \sum_{i=1}^{N} P_i\big(a_t(i)\big) \qquad (5)$$

III. Algorithm Analysis

Section III describes the learning pipeline used in the experiments. An offline dataset of simulator-generated transitions (s_t, a_t, r_t, s_{t+1}) is first constructed from the network model described in Section II. These transitions are used to train an enhanced dueling double DQN that outputs one binary activation decision per BS. During inference, the current network state is fed to the trained network, the binary action vector is obtained, and the minimum-active constraint is enforced through the feasibility projection described in Section II. The resulting policy is evaluated against three baselines: Always-On, Threshold-based, and Intelligent Heuristic. The overall information flow and interaction between learning, prediction, and allocation components are illustrated in Figure 1. Figure 1 should be read from left to right: the simulator provides the current network state and runtime features; the learning block maps this state to BS activation decisions; the allocation block applies the selected action after feasibility correction; and the KPI/logging block returns the resulting performance indicators for training and evaluation.

The training relies on an offline dataset generated from the simulator, where each entry captures the network state, the chosen action, the resulting next state, and the reward. A Double DQN is used, helping the model learn both the value of each

network state and the benefit of each action while reducing over-confident value estimates. Target-network updates, data scaling, and a structured training routine are applied to keep learning stable. To judge performance, the RL policy is compared against three reference strategies: keeping all cells active, a simple threshold rule, and a utility-based heuristic. Tests cover heavy traffic periods, low-traffic hours, and mixed daily patterns. For each scenario, multiple runs are executed to avoid one-off results. Performance is measured using several indicators, including energy use, delay, SLA violations, energy efficiency, and the number of active cells. BSs selected as available but receiving no UE traffic during that decision epoch are treated as idle, whereas available BSs serving traffic are treated as active. Although some BSs may be placed in sleep mode, QoS is preserved through three complementary mechanisms. First, after each control decision, UEs are re-associated to the BSs that remain available. Second, the action vector is constrained so that at least a minimum number of BSs remain active, which prevents overly aggressive deactivation and preserves basic coverage. Third, the reward function penalizes latency increases and SLA violations, so actions that reduce energy consumption at the expense of unacceptable service quality are disfavored during learning. Therefore, BS deactivation is not performed arbitrarily, but only when the remaining active infrastructure can absorb the traffic demand with acceptable performance. Results are summarized using averages and variation across runs to show both typical and worst-case behavior. Basic statistical checks are also applied to confirm that any improvements are consistent rather than random.

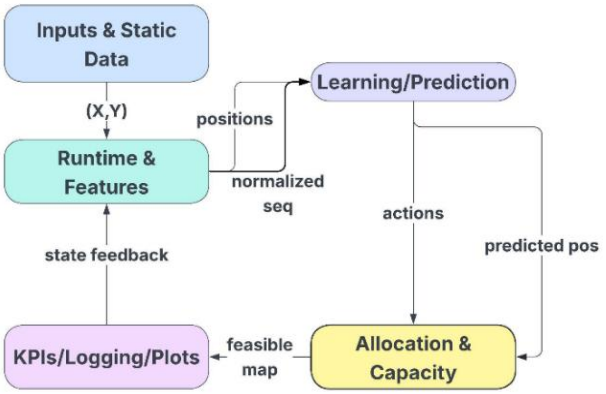

Fig. 1. Proposed Framework.

Algorithm 1 RL Dynamic Energy Management Algorithm.

Step 1: Problem Definition: The controller decides, at each time-step, which BSs stay active, idle or go to sleep so that total energy is reduced while latency, and an SLA latency remain acceptable.

Step 2: Parameters and Thresholds: The framework fixes BS power models (macro/small, active/idle/sleep), RL hyperparameters, QoS weights, time-step, area size, load thresholds for low/high demand, an SLA latency target, and a minimum number of active cells.

Step 3: Heterogeneous Topology Construction: One macro BS is placed at the center of the service area, and several small cells are positioned on a ring with mild randomness. Each BS has a coverage radius, a location, and an active power budget.

Step 4: Time-of-Day Traffic Profile: A daily pattern generates bounded traffic intensity with morning, lunch, and evening peaks and a night valley. This signal drives load pressure over time.

Step 5: Interference Map: For a given active/idle/sleep pattern, a distance-based function fills an inter-cell interference matrix. Contributions from far cells are small and values are capped for stability.

Step 6: Per-Cell State Vector: Each cell keeps six elements: load, latency, energy efficiency, channel quality, and a binary status flag. All values are clipped to the [0,1] range when needed.

Step 7: Transition Dynamics: Given the current state, actions, traffic intensity, and interference, the next state is computed. Sleeping lowers energy efficiency and raises latency; active cells evolve according to interference and channel quality.

Step 8: Reward Function: The scalar reward combines an energy term (penalizing high power draw) with QoS terms, average latency, a soft penalty for SLA violations, an energy-efficiency bonus, and a penalty when too few cells are active.

Step 9: Offline Data Generation: A large synthetic dataset is produced by rolling the transition model: sample a state, pick a heuristic action that respects thresholds and the minimum-active rule, compute reward and next state, and store (s,a,r,s) for all cells.

Step 10: Train/Validation Split and Scaling: The dataset is split into training and validation parts. A min–max scaler is fit on training features and applied to both current and next states to keep the same scale during learning and testing.

Step 11: Dueling Q-Network Design: A shared neural backbone maps the state to a latent vector. Two heads produce a state value and per-cell advantages for the two actions. They are combined into Q-values for each cell's active/sleep decision.

Step 12: Double DQN Targets and Optimization: The online network selects next actions; the target network evaluates them. The loss is the squared TD error between target returns and current Q-values for the logged actions. The target network is updated at fixed intervals, with learning-rate decay and gradient clipping.

Step 13: Policy Extraction: At inference, the normalized state passes through the network, and each cell chooses the higher-valued action. If fewer than the required minimum would be active, the highest-load cells are switched on to meet the constraint.

Step 14: Baseline Policies: Three baselines are prepared: all cells are always active; a threshold rule that activates cells above a load limit; and a utility heuristic that ranks cells by load, latency, energy efficiency, and a macro bonus.

Step 15: Simulator for Evaluation: A simulator initializes a realistic state and advances time by applying the chosen policy, computing interference, energy, and the next state. It logs energy, latency, SLA proxy, active cells, and reward at each step.

Step 16: Scenario Design and Runs: Three scenarios are tested: peak hour, off-peak, and variable start times. For each policy and scenario, multiple long episodes are executed to reduce randomness and capture steady behavior.

Step 17: Metric Aggregation: For every policy–scenario pair, the framework aggregates episode statistics: mean, standard deviation, minimum, and maximum of energy, latency, SLA violations, energy efficiency, and active cells.

Step 18: Statistical Comparison: When available, energy savings of the RL policy relative to each baseline are computed per scenario to quantify improvements in clear percentage terms.

Step 19: Visualization and Reporting: Bar plots with error bars compare policies across scenarios for all key metrics, training curves show convergence, and a summary table lists per-scenario means. Model files and metadata (normalization, topology) are saved for reuse.

Step 20: Safeguards and Constraints: The pipeline enforces a minimum number of active cells, caps interference, clips features, and uses stable training

settings. These measures prevent unrealistic actions and improve learning stability.

The implemented algorithm begins with a custom DataLoader module that ingests the offline ns-O-RAN dataset of over 300,000 transition tuples, shuffles them, and applies min–max normalization to each key performance metric feature for stable network training [6]. Batches of these normalized tuples (st, at, rt, st+1) are converted into TensorFlow tf.data pipelines, ensuring efficient GPU or CPU streaming during training [7]. The core QNetwork comprises two fully connected hidden layers with 256 and 128 ReLU-activated units, followed by an output layer of size 2N to represent binary RF-toggle actions for N Base Stations (BS). An identical TargetNetwork is maintained, with its parameters θ- synchronized from the online network θ every 1,000 steps to stabilize the bootstrap targets. During each training iteration, a mini-batch of 64 transitions is sampled uniformly from the replay buffer, and the temporal-difference target:

$$y = r + \gamma \max_{a'} Q(s', a'; \theta^{-}), \quad \gamma = 0.99 \qquad (5)$$

is computed to balance immediate and future rewards. The online network minimizes the mean-squared TD loss:

$$L(\theta) = E_{(s,a,r,s')\sim\mathbb{D}}[y - Q(s, a; \theta)]^2 \qquad (6)$$

via the Adam optimizer (learning rate 1×10^{-3}), following established best practices for DQN stability [8]. Double DQN is activated by selecting the maximizing action under θ and evaluating it under θ^{-}, effectively reducing overestimation bias [9]. In parallel, the Dueling DQN variant splits the final network layer into separate value V(s) and advantage A(s,a) streams and recombines them as:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s, a')\right) \qquad (7)$$

which speeds convergence in environments where many actions exhibit similar values [10]. For multi-scale control experiments, a FeUdal Network manager issues abstract subgoals every k steps, which a worker network translates into primitive RF-toggle actions, enabling efficient long-horizon credit assignment [11]. Trained models are serialized via ONNX and packaged into an O-RAN xApp, delivering inference decisions on the Near-RT RIC with sub-millisecond latency. Evaluation runs 100 simulator episodes, recording percentage energy reduction, and 95th-percentile latency, with paired t-tests ($p < 0.05$) verifying statistically significant gains over heuristic baselines. Hyperparameter tuning begins with 200 trials of Random Search over discount factor, learning rate, and ε-decay, then applies Bayesian Optimization via scikit-optimize and an Expected Improvement acquisition function to refine top configurations efficiently. Comprehensive profiling tracks replay buffer capacity, target-network update intervals and ε-greedy schedules, confirming that asynchronous A3C variants can reduce wall-clock training time by up to 50 % without impairing convergence demonstrating the algorithm's viability for real-time O-RAN deployment.

IV. Simulation Environment

The simulation environment was designed to emulate a heterogeneous 5G/6G cellular network with both macro and small cells operating under realistic traffic patterns. The environment integrates spatial topology generation, user distribution, traffic load evolution, and power consumption modeling to provide a representative testbed for reinforcement learning–based energy optimization. A lightweight Python simulator with a TensorFlow agent was implemented and used. Time is discrete; evaluation runs 30 episodes per scenario with 7,200 steps per episode. The offline dataset used for training is generated synthetically by rolling the same simulator. The network topology consists of seven BSs, including one macro cell at the center of the deployment area and six small cells distributed radially around it. The macro cell provides wide-area coverage with higher transmission power (400W), while the small cells are designed for localized coverage with lower power consumption (80W). Each BS is characterized by its type, coverage radius, transmit power in active mode, and geographic coordinates. Figure 2 illustrates the spatial topology of deployment, including the positions of macro and small cells.

A fixed deployment area of 2000 × 2000 meters was considered, within which User Equipments (UEs) were randomly distributed across the coverage region. In each simulation episode, up to 200 users were generated, each associated with the nearest base station according to distance and channel quality. The user distribution was updated dynamically during the simulation to reflect traffic mobility patterns. Figure 2 shows an example of UE placement relative to the deployed BS.

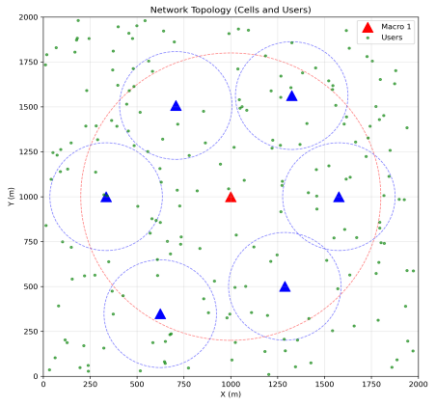

Fig. 2. Network Topology.

Traffic demand was modeled using a time-of-day dependent traffic profile, capturing realistic variations such as morning peaks, lunchtime bursts, and evening high-demand periods, as well as reduced night-time loads. This ensured that the learning agent experienced different operating conditions, including high-load stress and low-load energy-saving opportunities. The traffic profile was scaled to individual BS based on their cell type and coverage radius, leading to higher average load for the macro cell and lower but more variable load for the small cells. The power model uses four states: Active (macro) 400 W, Active (small) 120 W, Idle (macro) 80 W, Sleep (macro) 15 W. The per-cell state vector contains six normalized elements—load, throughput, latency, energy-efficiency, channel-quality, and a binary active/sleep flag. However, a BS that is kept available may operate as active when serving UEs or as idle when no UE is currently associated with it. A minimum of 2 active cells is enforced, and a normalized SLA latency threshold of 0.1 shapes the reward. To capture inter-cell interference, a matrix-based model was employed where interference was inversely proportional to the distance between active BS. This interference directly affected user latency, and energy efficiency, allowing the reinforcement learning agent to adaptively account for QoS degradation when making cell

activation decisions. The power consumption model considered four operating states for each base station:

TABLE 1. POWER CONSUMPTION PER OPERATING STATE

Operating State	Power (W)
Active (macro)	400 W
Active (small)	120 W
Idle (macro)	80 W
Sleep (macro)	15 W

The optimization of energy efficiency is achieved using reinforcement learning. The proposed method employs an Enhanced Dueling Double DQN, which extends the traditional Q-learning framework. In this approach, the dueling architecture allows the network to estimate separately the value of being in a given state and the advantage of selecting a particular action. In parallel, the double Q-learning mechanism mitigates the problem of overestimating Q-values, thus improving the stability of the learning process. By combining these mechanisms, the agent is able to identify effective policies for switching BS between active and sleep states while balancing quality of service and energy consumption.

V. PERFORMANCE EVALUATION

The proposed reinforcement learning–based framework was evaluated against three baseline strategies: Always-On, Threshold-based, and Intelligent Heuristic. Performance was assessed under three representative traffic scenarios—peak-hour demand, off-peak operation, and variable load conditions. Each scenario was simulated over thirty independent episodes, with each episode lasting 7,200 time steps. Figures 3 and 4 report the mean-squared temporal-difference (TD) loss defined in (6), averaged over the training and validation splits, respectively. As shown in Figure 3, the training loss decreases in a consistent manner across the training horizon and stabilizes near the end, reaching a final value of 7.15. This trend indicates that the optimization procedure is stable and that the network parameters converge toward a solution that fits the training experience.

Fig. 3. Training Loss of the RL Model.

Figure 4 reports the validation TD loss on held-out transitions. Unlike supervised-learning validation error, this quantity is not by itself a direct measure of final policy quality in offline RL, because it is sensitive to distribution shift between the logged training tuples and unseen traffic realizations. For this reason, Figure 4 is used here as a diagnostic indicator of value-function fit rather than as the primary criterion for policy selection. The increase in validation TD loss after approximately 30,000 steps suggests that the value approximator becomes less accurate on held-out transitions as training progresses. We therefore avoid interpreting this curve as evidence of improved generalization. Instead, the effectiveness of the learned policy is assessed through rollout-based evaluation metrics energy consumption, latency, SLA violations, and average active BSs reported in Figures 5 and 6 and in Table 2.

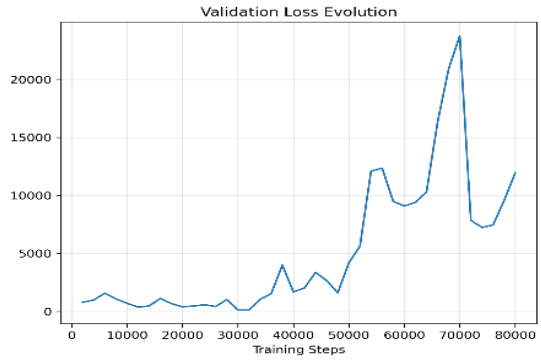

Fig. 4. Validation Loss of the RL Model.

Despite the volatility observed in the validation curve, the overall training process still converges with a policy that performs reliably during evaluation. In particular, the loss stabilization on the training set combined with the ability to maintain performance under diverse traffic realizations suggests that the learned policy does not merely memorize specific trajectories but captures patterns that transfer to unseen episodes in testing. Latency performance is depicted in Figure 5. The Always-On and Intelligent Heuristic approaches achieved near-baseline latency values of around 1.0, as expected due to the constant availability of all cells. The Threshold policy slightly degraded latency to about 0.993 because of occasional load concentration on fewer active cells. The Enhanced RL method, despite aggressively deactivating cells to conserve energy, maintained an average latency of approximately 0.893, which remains well within acceptable limits for delay-sensitive applications. This result indicates that the policy effectively balanced energy efficiency with quality of service.

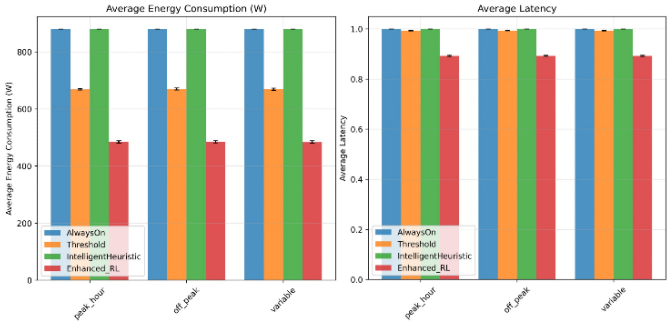

Fig. 5. Energy Consumption and Average Latency per Model.

Energy consumption results, presented in Figure 5, highlight the clear benefits of the reinforcement learning policy. The Always-On and Intelligent Heuristic approaches maintained all cells in active state, leading to an average consumption of 880 W across all scenarios. The Threshold policy provided moderate savings, reducing consumption to approximately 670 W by selectively deactivating cells under light load. In contrast, the Enhanced RL strategy achieved a substantial reduction of nearly 45% compared with Always-On and 27% compared with Threshold, lowering average consumption to about 484 W. These improvements were consistent across all three traffic scenarios, demonstrating the robustness of the learned policy. SLA violations, shown in Figure 6, provide further insight into

the trade-offs. The Always-On and Intelligent Heuristic strategies exhibited the highest violation counts, around 6,146 per episode, due to their inability to adapt resources dynamically to traffic variations. The Threshold method achieved minor improvements but still recorded more than 6,070 violations per episode. The Enhanced RL approach significantly reduced SLA violations to about 5,090 per episode, representing an improvement of approximately 17% compared with the baselines. This reflects the ability of the reinforcement learning policy to anticipate traffic conditions and allocate capacity more intelligently.

The analysis of average active cells (Figure 6) confirms the operational differences between the policies. As expected, both Always-On and Intelligent Heuristic strategies always maintained all seven cells active. The Threshold policy reduced this number to about four, achieving partial savings. In contrast, the Enhanced RL policy activated only two cells on average, yet maintained latency within target levels, highlighting its efficiency in exploiting spatial and temporal traffic variations.

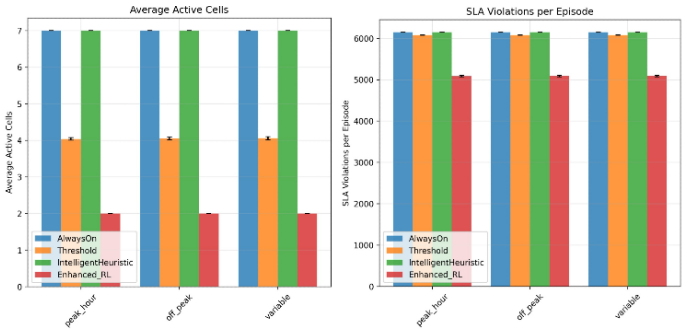

Fig. 6. SLA Violations and Average Active Cells per Model.

TABLE 2. SUMMARY OF THE RESULTS.

Policy	Avg. Energy (W)	Avg. Latency	SLA Violations (per episode)	Avg. Active Cells
Always-On	880.0	0.9999	6147	7.0
Threshold	669.2	0.9931	6079	4.05
Intelligent Heuristic	880.0	0.9999	6147	7.0
Enhanced RL	484.3	0.8928	5089	2.0

Table 2 highlights the clear operational advantages of the Enhanced RL method compared with all baseline strategies. While the Always-On and Intelligent Heuristic policies always maintain seven active cells, resulting in the highest energy footprint of 880 W and offering no adaptability, the Threshold method provides moderate savings by reducing activity to around four cells, yet still suffers from delayed reactions to sudden traffic variations. In contrast, the RL agent consistently operates with only two active cells on average, reaching the minimum allowed configuration and achieving the lowest energy consumption at 484.3 W, a reduction of roughly 45% relative to Always-On. Despite this aggressive reduction, the agent maintains competitive latency and significantly fewer SLA violations, demonstrating its ability to anticipate demand changes and activate resources only when they are genuinely required. The RL policy's advantage becomes particularly evident under fluctuating load conditions, where it avoids the overshooting and undershooting typical of static or threshold-based methods. By learning the interplay between load, interference, and future QoS impact, the agent delivers more stable performance across episodes, reducing unnecessary toggling and maintaining service quality even with minimal resource usage. Overall, Table 2 confirms that the Enhanced RL method achieves a near-optimal balance between energy savings and QoS compliance, outperforming static and heuristic strategies in terms of adaptability, robustness, and operational efficiency.

Overall, the results demonstrate that the proposed reinforcement learning–based approach delivers consistent and substantial energy savings while preserving service quality. The improvements were robust across traffic scenarios and were achieved without sacrificing latency or SLA compliance. By selectively activating only the necessary cells, the Enhanced RL method outperformed static and heuristic baselines in terms of both efficiency and adaptability.

VI. CONCLUSION AND FUTURE WORK

This paper studied the problem of reducing energy consumption in modern wireless networks through reinforcement learning methods. A simulation environment was designed to reflect the main characteristics of 5G and beyond, including heterogeneous BS deployments, realistic variations in traffic load, and inter-cell interference. Within this environment, a dueling double deep Q-learning approach was applied to balance the trade-off between service quality and energy efficiency. The results showed that the proposed method achieves significant energy savings compared with baseline policies, while keeping service performance at acceptable levels. These outcomes suggest that reinforcement learning can provide a practical basis for more sustainable operation of future mobile systems. At the same time, the work leaves open several points for further investigation. The current environment abstracts certain physical details such as beamforming, mobility, and channel estimation, which are important in real deployments. Extending the model to include these aspects would make the results more representative of actual networks. Another point concerns the generalization of the learning policy. While the method performed well under the tested conditions, its behavior under highly dynamic settings, such as rapid user movement, sudden surges in demand, or dense small-cell deployments, remains an open question.

Future studies should also consider how such reinforcement learning–based solutions could be deployed in practice. Real-time decision making, computational cost, and compatibility with current network standards are challenges that need to be addressed. In addition, collaborative approaches, such as distributed or federated reinforcement learning, may allow multiple BS to coordinate while limiting communication overhead. Finally, energy efficiency should not be treated in isolation, but as part of a broader set of goals that also include reliability, security, and sustainability, especially as research moves toward the design of 6G systems. In conclusion, the study shows that reinforcement learning can play an important role in improving the energy performance of wireless networks. Building on this foundation, future research can develop models that are both more realistic and more adaptable, helping move closer to networks that are efficient, reliable, and sustainable.

REFERENCES

[1] A. El Amine, J.-P. Chaiban, H. Al Haj Hassan, P. Dini, L. Nuaymi, and R. Achkar, "Energy optimization with multi-sleeping control in 5G heterogeneous networks using reinforcement learning," IEEE Trans. Netw. Serv. Manag., vol. 19, no. 4, pp. 4313–4327, Dec. 2022.

[2] H. Zhou et al., “Hierarchical reinforcement learning for RIS-assisted energy-efficient RAN,” arXiv preprint arXiv:2301.02771, Jan. 2023.

[3] S. Bassoy, R. Behravesh, and J. Pujol-Roig, “SEEDRL: Smart energy efficiency using deep reinforcement learning for 6G networks,” in Proc. IEEE Globecom Workshops, Madrid, Spain, Dec. 2023, pp. 732–737.

[4] D. Wu et al., “Energy saving in cellular wireless networks via transfer deep reinforcement learning,” in Proc. IEEE GLOBECOM, Madrid, Spain, Dec. 2023, pp. 7019–7024.

[5] H. Park and Y. Lim, “Reinforcement learning for energy optimization with 5G communications in vehicular social networks,” Sensors, vol. 20, no. 8, p. 2361, Aug. 2020.

[6] M. Bordin et al., "Design and Evaluation of Deep Reinforcement Learning for Energy Saving in Open RAN," 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2025, pp. 1-6.

[7] Q. Wang, S. Chetty, A. Al-Tahmeesschi, X. Liang, Y. Chu and H. Ahmadi, "Energy Saving in 6G O-RAN Using DQN-Based xApp," 2024 IEEE 29th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Athens, Greece, 2024, pp. 01-06.

[8] G. Chen, R. Shao, F. Shen, and Q. Zeng, “Slicing Resource Allocation Based on Dueling DQN for eMBB and URLLC Hybrid Services in Heterogeneous Integrated Networks,” Sensors, 2023.

[9] M. Sewak, “Deep Q Network (DQN), Double DQN, and Dueling DQN”, Deep Reinforcement Learning, pp. 95-108, Springer, 2019.

[10] J. Ren and Z. Chai, “Joint Spectrum Allocation and Power Control in Vehicular Communications Based on Dueling Double DQN,” 2022.

[11] U. Keerthan, N. K. Musunuru, A. J, and K. G. Gurunadh, “Energy Management System – Deep Reinforcement Learning Based Dueling DQN (Deep Q-Networks),” in 2024 Asia Pacific Conference on Innovation in Technology (APCIT), IEEE, 2024.