International Journal of Innovative Information Systems & Technology Research 14(3):353-362, July-Sept, 2026

© SEAHI PUBLICATIONS, 2026 www.seahipublications.org ISSN: 2467-8562

DOI:10.5281/zenodo.22806950

Lightweight Compression of DQN-Based Energy Management for Resource-Constrained Edge-Fog-Cloud IoT Systems

[1]A. A. Okpo *[2]O. E. Taylor *[3]N. D. Nwiabu

[1,2&3]Department of Computer Science, Rivers State University, Port-Harcourt, Nigeria
Corresponding Author: Okpo, Augustine Akachi
Corresponding Author's Email: okpo.akachi@rsu.edu.ng

ABSTRACT
Deep reinforcement learning is increasingly used for adaptive energy management in edge-fog-cloud Internet of Things (IoT) systems, but a learned policy is useful at the edge only when its memory footprint, inference latency, and processing overhead remain small. This paper evaluates compression of a Deep Q-Network (DQN) energy-management policy for simulated edge-fog-cloud IoT environments. The original DQN selected composite control actions across dynamic voltage and frequency scaling, task offloading, and duty cycling from a 13-variable state representation covering workload demand, device capacity, link condition, and energy status. The trained policy was compressed with 50 percent unstructured magnitude pruning and dynamic int8 post-training quantisation of linear layers. Evaluation compared the uncompressed and compressed DQN using model size, reward retention, inference latency, computational overhead, energy cost, latency cost, and feasibility violations. Public topology data from Internet Topology Zoo and CAIDA were combined with generated workload, device-profile, link-profile, and energy-state traces to provide reproducible simulation inputs. Compression reduced model size from 0.0284 MB to 0.0119 MB, a reduction of about 58.1 percent and a compression ratio of about 2.39:1. Reward retention ranged from 100.0499 percent to 100.1327 percent relative to the uncompressed DQN. The compressed policy maintained average computational overhead of 1.862 percent and inference latency of 21.3744 ms, below the 5 percent and 50 ms targets. It also preserved the energy-management benefit, with an average 28.010 percent energy improvement over static control. The results indicate that pruning and post-training quantisation can reduce deployment cost without reducing evaluated reward performance.

Keywords: Deep Q-Network; model compression; pruning; quantisation; edge computing; fog computing; Internet of Things; energy management; resource-constrained devices

I. INTRODUCTION

Edge-fog-cloud IoT systems distribute computation across sensors, edge gateways, fog servers, and cloud resources. This arrangement improves responsiveness and reduces unnecessary cloud dependence, but it also creates energy-management difficulty because devices differ in processing capacity, memory, battery state, communication cost, and latency tolerance [1], [2]. Energy-control mechanisms such as dynamic voltage and frequency scaling (DVFS), task offloading, and duty cycling can reduce energy consumption, but their effectiveness depends on selecting actions that fit the current workload, link condition, and device state [3], [4].

Deep reinforcement learning is suitable for this kind of adaptive decision-making because an energy-management controller can observe a state, select an action, receive feedback, and improve future action choices. Deep Q-Networks (DQN) are particularly useful when the action space is discrete, such as selecting among predefined DVFS levels, offloading destinations, and duty-cycle modes [5], [6]. However, the practical use of learned policies in constrained IoT and edge environments is limited by model size, inference latency, processing overhead, and memory pressure [7], [8]. A policy that saves energy in simulation may lose deployment value if the policy itself consumes excessive computation while making decisions.

DQN-based energy-management policies face a deployment problem in resource-constrained IoT environments. A policy may learn useful state-dependent energy-control decisions, yet still be unsuitable for edge use if its model size, inference latency, or processing overhead is too high. In this work, a trained DQN energy-management policy was compressed using magnitude pruning and dynamic int8 post-training quantisation. The compressed model was then compared with the uncompressed DQN to determine whether compression reduced model size while preserving reward performance, energy behaviour, overhead, and latency feasibility. Static control was included as a contextual baseline for energy improvement.

The remainder of the paper is organised as follows. Section II reviews related work on reinforcement-learning energy management and lightweight model compression. Section III presents the method. Section IV reports and discusses compression, reward retention, overhead, latency, and preserved energy performance. Section V concludes the paper.

II. Related Work

Energy-aware computing aims to reduce unnecessary power consumption by adapting computation, communication, and device activity to workload and operating conditions. In massive IoT networks, energy control must account for sensing, communication, and processing costs across heterogeneous devices [1]. In industrial IoT and green 6G environments, energy-aware communication and distributed resource control are also central to sustainable network operation [2]. Fog computing studies further identify workload placement, resource provisioning, and energy-aware scheduling as major concerns for sustainable distributed systems [3]. These studies show that energy management in IoT and fog systems must consider both computation and communication rather than treating energy as an isolated metric.

Static and heuristic energy-management methods remain attractive because they are simple and lightweight. Distributed energy-aware edge provisioning has been implemented through rule-driven resource allocation [4]. Tier-aware node selection has also been used in layered edge-fog-cloud IoT architectures, where task placement is guided by energy and communication conditions [9]. In wireless sensor networks, energy-efficient communication and routing schemes can extend network lifetime through cluster-head selection and energy-aware routing [10]. Improved ant colony routing has similarly linked path selection to residual battery level and link quality [11]. These works support the value of energy-aware control, but fixed rules and iterative heuristics may respond poorly when workload, topology, battery, or link conditions change.

Machine learning has been used to make energy management more adaptive. In LoRaWAN energy optimisation, artificial intelligence and machine-learning methods support protocol-parameter adjustment from observed network conditions [12]. In heterogeneous fog systems, workload forecasting and scheduling can improve energy and latency performance, although prediction introduces additional overhead [13]. Enhanced fuzzy logic has also been applied to dynamic fog task allocation, showing that lightweight reasoning can support real-time allocation decisions [14]. These approaches improve on fixed control, but their deployment cost still matters when inference is performed near constrained edge devices.

Reinforcement learning provides a stronger adaptation mechanism because policies can learn from repeated interaction with the environment. In vehicular edge-computing task offloading, deep reinforcement learning has been used to balance local computation, wireless transmission, and cloud-

processing delay [5]. In smart-home energy management, adaptive reinforcement learning enables scheduling decisions to respond to occupancy and electricity-price conditions [6]. Federated reinforcement learning extends adaptive resource allocation to distributed edge-IoT systems, but it also introduces coordination and inference overhead [15]. Hybrid reinforcement-learning and metaheuristic methods have also been used for wireless sensor network energy efficiency, improving energy behaviour while increasing training and coordination demands [16]. These studies support reinforcement learning for adaptive energy management, but they also reinforce the need to measure the cost of running the learned policy.

Lightweight deep learning techniques address this deployment burden by reducing model size and inference cost. Pruning, quantisation, knowledge distillation, and hardware-aware optimisation are commonly identified as core strategies for resource-constrained learning [7]. Hybrid deep-learning architectures for IoT can provide strong task performance, but their inference latency and memory demand illustrate how neural models may become expensive for constrained devices [8]. Lightweight machine-learning evaluations in constrained IoT security also show the importance of measuring computational overhead alongside task performance [17]. Quantum-inspired optimisation for edge task offloading further shows that optimisation quality must be considered together with computational cost [18].

Taken together, the literature points to a deployment gap. Energy-aware IoT systems increasingly need adaptive control, and reinforcement learning is a strong candidate for state-dependent decisions. At the same time, the learned policy must be small and fast enough to justify use near constrained edge devices. Existing adaptive studies usually emphasise energy savings, while lightweight-learning studies often treat compression outside a concrete energy-management policy. The present work connects both sides by compressing a DQN-based IoT energy-management policy and measuring whether reward and operational feasibility are retained under simulated edge-fog-cloud constraints.

III. METHOD

Figure 1 summarises the architecture used for the compression experiment. Public topology data and generated operating traces were converted into model-ready state records. These records trained an uncompressed DQN energy-management policy, after which the trained policy was passed through pruning and post-training quantisation. The compressed and uncompressed policies were then evaluated on the same records. Static control was retained only as a reference for checking whether the compressed policy still produced energy savings relative to fixed control.

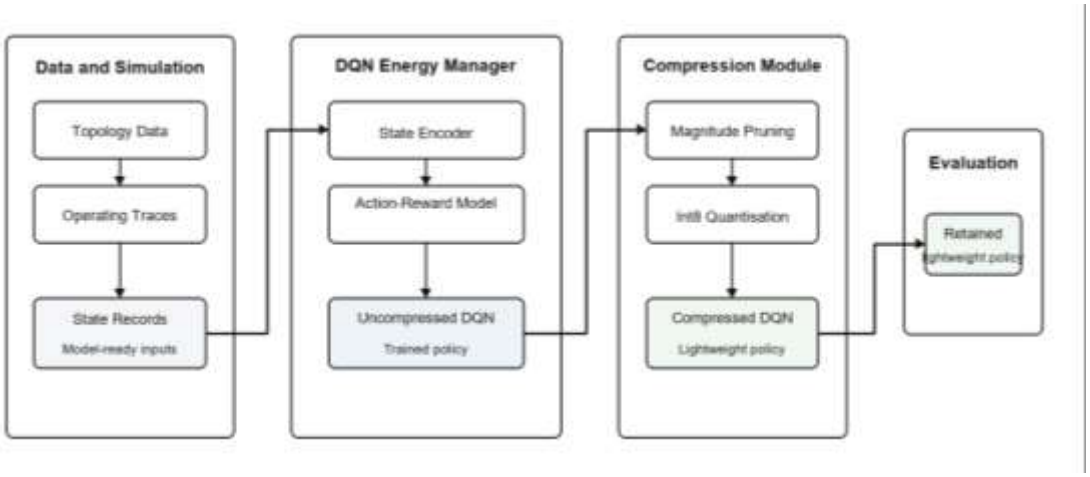

Figure 1. DQN-based energy-management policy compression architecture

Simulation Inputs

The simulation used public topology data from Internet Topology Zoo and CAIDA. Internet Topology Zoo GraphML files provided real graph structures for the smaller scenarios, while CAIDA AS-relationship data supported the largest metropolitan-scale scenario. The topologies were combined with generated device-profile, link-profile, workload, and energy-state traces. Table 1 summarises the loaded graph structures.

Table 1: Simulation Configurations and Loaded Topology Sizes

Configuration	Target Nodes	Loaded Nodes	Loaded Edges	Workload Pattern	Network Source
Small-Scale Edge	100	19	24	Periodic 24-hour cycle	Internet Topology Zoo
Medium-Scale Fog	500	53	89	Bursty Poisson arrivals	Internet Topology Zoo
Large-Scale Hierarchical	1,000	158	189	Mixed periodic and bursty traffic	Internet Topology Zoo
Metropolitan IoT	5,000	5,000	18,318	Steady-state with peak demand	CAIDA and Topology Zoo

The Topology Zoo scenarios loaded fewer nodes than their target labels because the selected public GraphML files contained smaller connected graph structures. The CAIDA-based Metropolitan IoT scenario reached the 5,000-node target and provided the largest evaluation graph. Across the four configurations, the preprocessing stage produced 3,909 training records, 1,001 validation records, and 807 evaluation records.

DQN Policy

The DQN policy selected composite energy-control actions from a 24-action space. Each action combined one DVFS level, one offloading destination, and one duty-cycle mode, as shown in Table 2.

Table 2: Action Space Definition

Action Dimension	Options	Count
DVFS level	Low, medium, high	3
Offloading destination	Local, edge, fog, cloud	4
Duty-cycle mode	Eco, active	2
Composite actions	DVFS x offload x duty cycle	24

The state vector contained 13 normalised variables:

$$s_t = [task_size,\ cpu_cycles,\ deadline,\ processing_capacity,\ memory,\ power_budget,\ base_power,\ ba \quad (1)$$

These variables represented workload demand, device capacity, link condition, and energy state. They supported eight operational constraints: energy consumption, latency, computational overhead, bandwidth availability, memory availability, processing capacity, battery level, and task deadline.

The reward was based on a weighted cost:

$$C = 0.45E + 0.25L + 0.20O + 0.10V \quad (2)$$

where C is the weighted cost, E is energy cost, L is latency cost, O is computational-overhead cost, and V is the feasibility-violation count. The reward was computed as:

$$R = \frac{1}{1 + weighted_cost} \tag{3}$$

The DQN used two hidden layers of 64 units. Training used 70 episodes, 80 steps per episode, a learning rate of 0.001, a discount factor of 0.92, an initial epsilon of 1.0, and a minimum epsilon of 0.05.

Model Compression

The trained DQN was compressed in two stages. First, 50 percent unstructured magnitude pruning was applied to the weights of the linear layers. Low-magnitude weights were removed because they were treated as lower-contribution parameters in the trained network. Second, dynamic int8 post-training quantisation was applied to the pruned linear layers, reducing numerical precision during inference. The compressed policy kept the same state vector and 24-action decision space as the uncompressed policy, so the comparison measured compression of the learned model rather than a change in the control problem.

Evaluation Metrics

The primary compression metric was model-size reduction:

$$Size\ Reduction = \frac{S_{uncompressed} - S_{compressed}}{S_{uncompressed}} \times 100\% \tag{4}$$

where $S_{uncompressed}$ is the uncompressed DQN size and $S_{compressed}$ is the compressed DQN size.

Reward retention measured whether compression preserved policy quality:

$$Reward\ Retention = \frac{R_{compressed}}{R_{uncompressed}} \times 100\% \tag{5}$$

where $R_{compressed}$ and $R_{uncompressed}$ are the mean rewards of the compressed and uncompressed DQN policies in the same scenario.

Computational overhead was estimated as:

$$OH = \frac{FLOPs_{inference}}{FLOPs_{capacity}} \times 100\% \tag{6}$$

The evaluation also reported inference latency, energy cost, latency cost, overhead cost, feasibility violations, and energy improvement over static control. The target thresholds were model-size reduction without reward loss, reward retention above 95 percent, computational overhead below 5 percent, and inference latency below 50 ms.

Software and Hardware Environment

The system was implemented in Python 3.10 using NumPy, Pandas, NetworkX, scikit-learn, PyTorch CPU, Matplotlib, Seaborn, and Jupyter [19]. Experiments were executed on a Windows 10 system with four logical CPU cores and 11.95 GB RAM. No GPU was required. A fixed random seed of 42 supported reproducible trace generation, data splitting, DQN training, and policy evaluation. The study used public

topology datasets and generated simulation traces; no human-participant data, private user records, or live IoT deployment records were collected.

IV. RESULTS AND DISCUSSION

DQN Training

The DQN was first trained to provide the uncompressed policy used in the compression experiment. Table 3 gives the training summary. The final five reward values were 0.6043, 0.6119, 0.5763, 0.5974, and 0.6114, indicating that the later episodes remained within a narrow reward band despite continued exploration.

Table 3: DQN Training History Summary

Metric	Count	Mean	Standard Deviation	Minimum	Median	Maximum
Reward	70	0.5916	0.0502	0.1870	0.5977	0.6273
Loss	70	0.0345	0.0643	0.0068	0.0200	0.4670
Epsilon	70	0.2661	0.2498	0.0500	0.1619	0.9500

Model Size

Pruning and dynamic int8 quantisation reduced the DQN model from 0.0284 MB to 0.0119 MB, giving an approximate 58.1 percent reduction and a compression ratio of about 2.39:1.

Table 4: Compression Effect

Policy	Model Size (MB)	Size Reduction (%)	Compression Ratio
Uncompressed DQN	0.0284	0.0	1.00:1
Compressed DQN	0.0119	58.1	2.39:1

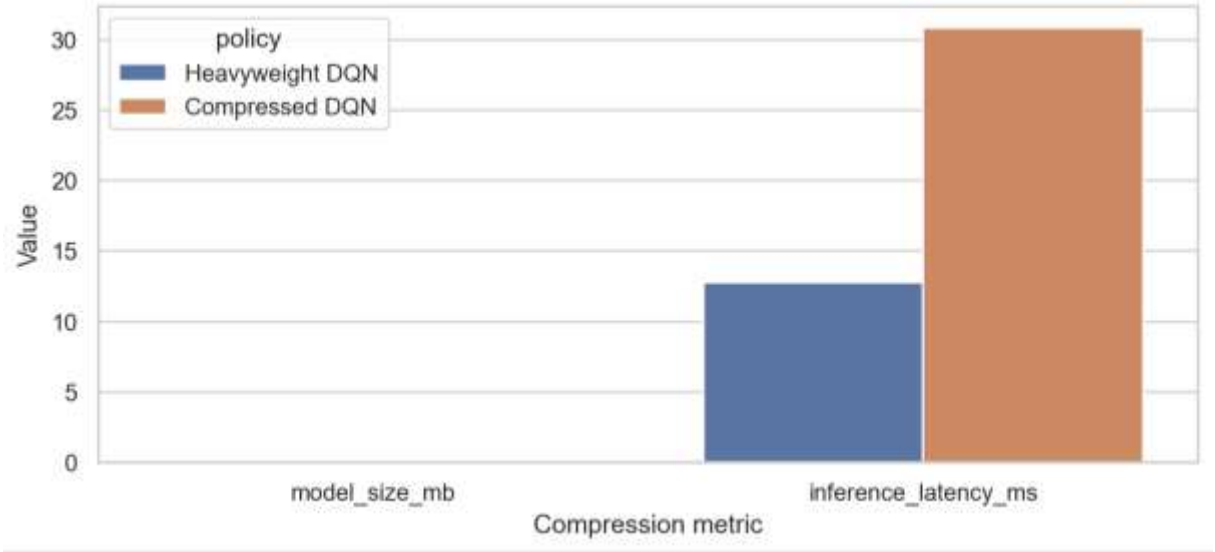

Figure 2. Model compression effect

The compressed model retained the same state representation and action space as the uncompressed DQN. The reduction therefore came from the policy representation itself, not from simplifying the energy-management problem.

Reward Retention

The main concern after compression was whether the smaller model still behaved like the learned policy. Table 5 compares reward values for the uncompressed and compressed DQN across the four configurations.

Table 5: Reward Retention of the Compressed DQN

Configuration	Uncompressed DQN Reward	Compressed DQN Reward	Reward Retention (%)
Large-Scale Hierarchical	0.6345	0.6352	100.1130
Medium-Scale Fog	0.6397	0.6404	100.1003
Metropolitan IoT	0.6341	0.6344	100.0499
Small-Scale Edge	0.6372	0.6380	100.1327
Average	0.6364	0.6370	100.0990

Reward retention exceeded the 95 percent target in every configuration. The compressed DQN slightly outperformed the uncompressed DQN on these evaluation records, but this should be read conservatively. The result indicates that pruning and quantisation did not weaken the evaluated policy; it does not establish that compression will generally improve DQN policies.

Overhead and Inference Latency

The compressed model also met the lightweight feasibility targets. Average computational overhead was 1.862 percent, below the 5 percent target, and mean inference latency was 21.3744 ms, below the 50 ms target.

Table 6: Compressed DQN Performance Against Lightweight Targets

Configuration	Overhead (%)	Inference Latency (ms)	Reward Retention (%)
Large-Scale Hierarchical	1.8828	21.3744	100.1130
Medium-Scale Fog	1.8044	21.3744	100.1003
Metropolitan IoT	1.8673	21.3744	100.0499
Small-Scale Edge	1.8926	21.3744	100.1327
Average	1.8618	21.3744	100.0990
Target	<5.0000	<50.0000	>95.0000

These values are central to the paper because they measure the cost of using the learned policy, not only the energy cost produced by the policy.

Energy Performance After Compression

Energy performance was retained after compression. Table 7 reports a compact scenario-level comparison, using static control only as the reference for energy improvement.

Table 7: Energy Performance After Compression

Configuration	Static Energy	Uncompressed DQN Energy	Compressed DQN Energy	Compressed Improvement (%)
Large-Scale Hierarchical	0.4612	0.3370	0.3275	28.9841
Medium-Scale Fog	0.4748	0.3563	0.3487	26.5675
Metropolitan IoT	0.4769	0.3430	0.3401	28.6759
Small-Scale Edge	0.4269	0.3143	0.3082	27.8107
Average	0.4600	0.3377	0.3311	28.0100

The compressed DQN recorded lower energy cost than the uncompressed DQN in all four configurations and maintained an average 28.010 percent improvement over static control. The policy therefore remained effective after its footprint was reduced.

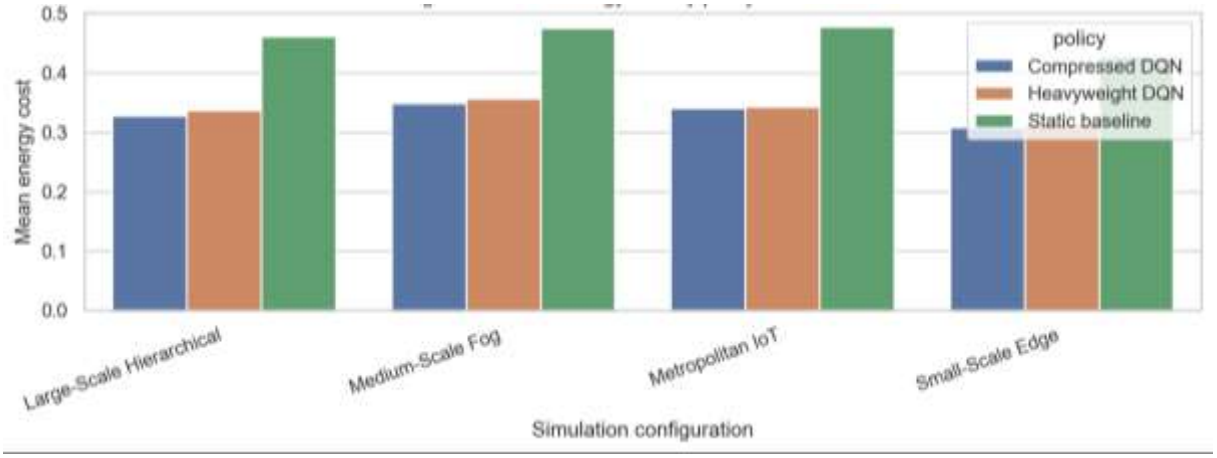

Figure 3. Mean energy cost by policy and scenario

Latency and Constraint Trade-Offs

The compressed DQN had higher simulated latency cost than static control because it sometimes selected energy-saving actions with higher task-latency cost. However, model inference latency stayed below the 50 ms threshold. The distinction matters: task-latency cost describes the simulated effect of selected actions, whereas inference latency measures the time needed to run the policy.

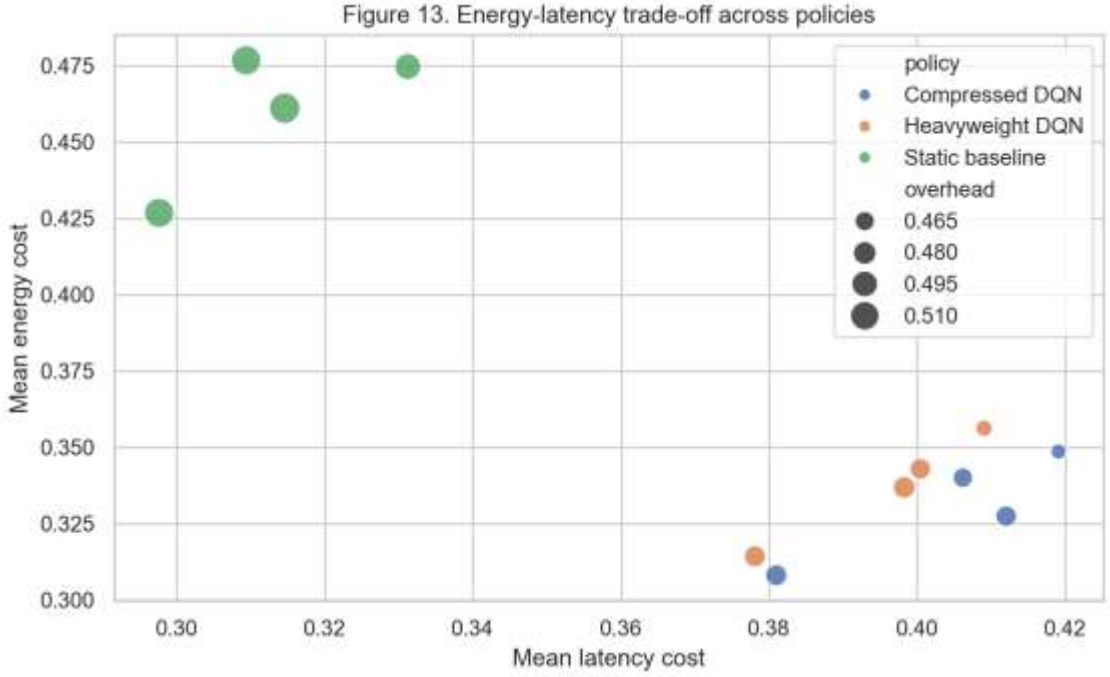

Figure 4. Energy-latency trade-off

The aggregate feasibility-violation score was also slightly lower for the compressed DQN than for static control, with average violations of 2.4262 and 2.4783 respectively. Thus, the compressed model reduced policy size and retained reward performance without worsening aggregate feasibility in the simulation.

The results remain bounded by the simulation setting. Workload, device, link, and energy-state records were generated rather than collected from live deployments, and the Topology Zoo scenarios used smaller connected graph structures than their target labels. Since the model was evaluated on CPU-only

commodity hardware rather than physical low-power IoT boards, the results should be interpreted as controlled simulation evidence rather than direct field deployment evidence.

V. CONCLUSION

Pruning and dynamic post-training quantisation reduced the footprint of the DQN-based energy-management policy in the simulated edge-fog-cloud IoT environment. Model size fell from 0.0284 MB to 0.0119 MB, corresponding to an approximate 58.1 percent reduction and a compression ratio of about 2.39:1. The compressed DQN retained reward performance across all scenarios, with reward retention between 100.0499 percent and 100.1327 percent relative to the uncompressed DQN.

The compressed policy also remained within the lightweight feasibility targets, with average computational overhead of 1.8618 percent and inference latency of 21.3744 ms. Its average energy improvement over static control was 28.010 percent, indicating that the energy-management behaviour of the learned policy was preserved after compression. Future work should validate the compressed policy on physical IoT and edge hardware and compare pruning and quantisation with other compression strategies such as structured pruning, knowledge distillation, mixed-precision quantisation, and hardware-aware optimisation.

The model can be improved in future work by evaluating it on physical IoT and edge-fog testbeds, adding real workload and battery traces, and testing it under stronger network-failure, mobility, and device-depletion conditions. Further improvement can also come from comparing DQN with Double DQN, Dueling DQN, actor-critic methods, federated reinforcement learning, genetic algorithms, and particle swarm optimisation. Additional compression methods such as structured pruning, knowledge distillation, mixed-precision quantisation, and hardware-aware optimisation should also be investigated to determine whether the model can achieve lower latency and overhead on real edge hardware.

REFERENCES

[1] M. H. Alsharif, A. H. Kelechi, J. Kim, and J. H. Kim, "A comprehensive survey of energy-efficient computing to enable sustainable massive IoT networks," *Alexandria Engineering Journal*, vol. 88, pp. 214-240, 2024.

[2] K. Fernando and G. Lazaroiu, "Energy-efficient industrial Internet of Things integration in green 6G networks," *Applied Sciences*, vol. 14, no. 18, Article 8558, 2024. https://doi.org/10.3390/app14188558

[3] M. H. Alsharif, A. H. Kelechi, Y. Yahya, and S. A. Chaudhry, "Survey of energy-efficient fog computing techniques and recent advances: Toward a roadmap for sustainable fog computing ecosystems," *Journal of Network and Computer Applications*, vol. 233, Article 104063, 2025.

[4] M. Avgeris, D. Spatharakis, D. Dechouniotis, A. Leivadeas, V. Karyotis, and S. Papavassiliou, "ENERDGE: Distributed energy-aware resource allocation at the edge," *Sensors*, vol. 22, no. 2, Article 660, 2022. https://doi.org/10.3390/s22020660

[5] W. Shi, L. Chen, and X. Zhu, "Task offloading decision-making algorithm for vehicular edge computing: A deep-reinforcement-learning-based approach," *Sensors*, vol. 23, no. 17, Article 7595, 2023. https://doi.org/10.3390/s23177595

[6] P. Saroha et al., "Dynamic appliance scheduling and energy management in smart homes using adaptive reinforcement learning techniques," *Scientific Reports*, vol. 15, no. 1, Article 9, 2025. https://doi.org/10.1038/s41598-025-08125-9

[7] Y. Liu, Y. Wang, and X. Li, "Lightweight deep learning for resource-constrained environments: A survey," *ACM Computing Surveys*, vol. 56, no. 9, Article 227, 2024.

[8] P. Sinha et al., "A high performance hybrid LSTM CNN secure architecture for IoT environments using deep learning," *Scientific Reports*, vol. 15, no. 1, Article 9684, 2025. https://doi.org/10.1038/s41598-025-94500-5

[9] R. Fereira, C. Ranaweera, K. Lee, and J. G. Schneider, "Energy efficient node selection in edge-fog-cloud layered IoT architecture," *Sensors*, vol. 23, no. 13, Article 6039, 2023. https://doi.org/10.3390/s23136039

[10] D. Gupta, S. Wadhwa, S. Rani, Z. Khan, and W. Boulila, "EEDC: An energy efficient data communication scheme based on new routing approach in wireless sensor networks for future IoT applications," *Sensors*, vol. 23, no. 21, Article 8839, 2023. https://doi.org/10.3390/s23218839

[11] H. Han, J. Tang, and Z. Jing, "Wireless sensor network routing optimization based on improved ant colony algorithm in the Internet of Things," *Heliyon*, vol. 10, no. 1, Article e23577, 2024. https://doi.org/10.1016/j.heliyon.2023.e23577

[12] M. Alkhayyal and A. Mostafa, "Recent developments in AI and ML for IoT: A systematic literature review on LoRaWAN energy efficiency and performance optimization," *Sensors*, vol. 24, no. 14, Article 4482, 2024. https://doi.org/10.3390/s24144482

[13] S. R. Hassan, A. U. Rehman, N. Alsharabi, S. Arain, A. Quddus, and H. Hamam, "Design of load-aware resource allocation for heterogeneous fog computing systems," *PeerJ Computer Science*, vol. 10, Article e1986, 2024. https://doi.org/10.7717/peerj-cs.1986

[14] W. Jin and A. Rezaeipanah, "Dynamic task allocation in fog computing using enhanced fuzzy logic approaches," *Scientific Reports*, vol. 15, Article 18513, 2025. https://doi.org/10.1038/s41598-025-03621-4

[15] S. Mali et al., "Federated reinforcement learning-based dynamic resource allocation and task scheduling in edge for IoT applications," *Sensors*, vol. 25, no. 7, Article 2197, 2025. https://doi.org/10.3390/s25072197

[16] S. Zhang and X. Liu, "Improving the functionality of wireless sensor networks through the use of reinforcement learning and metaheuristic based energy efficient system," *Scientific Reports*, vol. 15, no. 1, Article 30758, 2025. https://doi.org/10.1038/s41598-025-16128-9

[17] M. A. Alwaisi, H. M. Ghaleb, and M. Al-Shaher, "Securing constrained IoT systems: A lightweight machine learning approach for anomaly detection and prevention," *Internet of Things*, vol. 28, Article 101446, 2024.

[18] B. Naik, J. Nayak, H. S. Behera, and A. Abraham, "A quantum inspired particle swarm optimization approach for energy-efficient task offloading and resource allocation in edge computing," *Cluster Computing*, vol. 27, pp. 5975-5996, 2024.

[19] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. CreateSpace, 2009.