Sensors and Materials, Vol. 38, No. 5 (2026) 2753–2768
MYU Tokyo

S & M 4467

Optimizing Autonomous Robot Control Algorithm: Deep Reinforcement Learning for Dynamic and Uncertain Environments

Weiwei Li[1] and Chi Zhang[2*]

[1]Fuyang Institute of Technology, Fuyang 236031, China
[2]School of Artificial Intelligence and Big Data, Hefei University, Hefei 230601, China

(Received August 21, 2025; accepted May 13, 2026)

Keywords: autonomous robot, machine learning, real-time decision-making, algorithm optimization

The convergence of AI, machine learning, and sensor technologies has contributed to the advancement in robotics, enabling autonomous operation in complex and dynamic environments. Conventional control algorithms lack flexibility and adaptive decision-making, whereas deep reinforcement learning leverages multi-dimensional sensor data to learn complex behaviors. We developed and evaluated a Deep reinforcement learning (DRL)-based robot control system that operates under severe and uncertain conditions. The system integrated sensitive environmental sensor data with enhanced neural network architectures, including convolutional and recurrent layers, and employed a hybrid model-free/model-based agent design to balance learning efficiency with adaptability. Simulation and statistical analysis results demonstrated consistent learning patterns, with a running average reward stabilizing at −1177.28 (a standard deviation of 131.98), showing policy convergence. Evaluation across 50 episodes yielded a mean reward of −1333.46 (a standard deviation of 210.58), below the predefined success threshold of −200. These results show the developed system's ability to learn navigation behaviors while underscoring the need for further optimization. The architecture contributes to sensor technology by enabling active sensing, multimodal integration, and compensation for material-level nonlinearities, supporting the development of cost-efficient, sustainable sensors and advancing reliable, adaptive autonomous robotic control systems.

1. Introduction

The integration of AI, machine learning (ML), and advanced sensor technologies has rapidly advanced robotics technology. Such advancements have enabled autonomous robots to be increasingly deployed in industrial automation, healthcare, autonomous vehicles, and disaster response, where they must operate under complex and dynamic conditions.(1) However, industrial applications of the robots with conventional robot control algorithms are challenging owing to limitations in flexibility, operational inefficiency, and adaptability.(2) To address these limitations, next-generation control algorithms are required, emphasizing learning, adaptation,

*Corresponding author: e-mail: hfuu_zc@hfuu.edu.cn
https://doi.org/10.18494/SAM5898

ISSN 0914-4935

and rapid autonomous decision-making.[3] Deep reinforcement learning (DRL), an extension of reinforcement learning, enables complex behaviors of robots through sequential decision-making. Whereas supervised learning requires labeled data, DRL enables direct learning from high-dimensional sensory inputs by interacting with the environment and receiving rewards or penalties (Fig. 1).[4,5] Therefore, DRL has been widely applied to game playing, robotic manipulation, and autonomous navigation.[6] However, its application in robotics is constrained by computational inefficiency, safety risks, and difficulties in transferring simulated results to real-world operations.[7]

A major challenge in DRL application to robotic control is the handling of sensor data under real-world conditions, since sensor data are often incomplete, noisy, or delayed, particularly in autonomous navigation (Fig. 2).[8] Dynamic environments with fluctuating boundary conditions, diverse objectives, and unexpected disturbances further complicate learning from sensor data, leading to suboptimal or unsafe behaviors of robots.[9] To address such issues in using DRL in robotics, recurrent neural networks and attention mechanisms have been incorporated to capture temporal dependencies and improve performance under partial observability.[10]

Another limitation of using DRL in robotics is the extensive data requirements of DRL agents (autonomous computational entities such as robot brains that observe the environment, make decisions, and learn from feedback). Training deep neural networks demands large volumes of sensor data, which can be costly, time-intensive, and hazardous to collect.[11,12] Therefore, it is important to improve sensing efficiency through domain randomization and

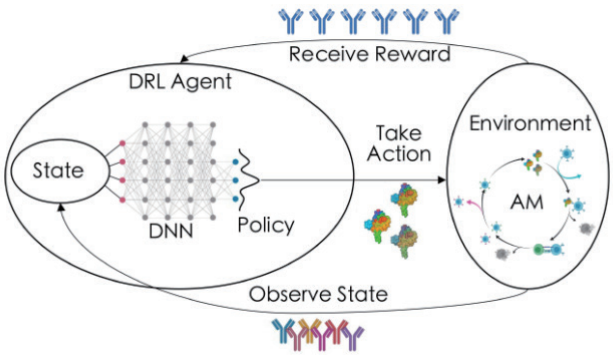

Fig. 1. (Color online) Representative structure of DRL.[7]

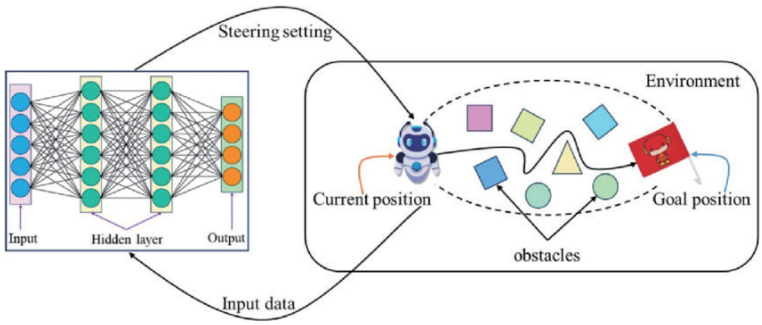

Fig. 2. (Color online) Structure of DRL-based robot control system.[10]

transferring simulation results to real-world operations.[13] Nonetheless, ensuring generalization and stability under uncertainty remains a pressing challenge.

DRL is also applied to algorithmic optimization for the performance of sensor networks. Traditional sensors often fail to conform to complex surfaces, limiting their utility in adaptive robotics. Materials such as liquid metals or piezoresistive elastomers are used in sensors, offering high sensitivity, but those sensors suffer from hysteresis and nonlinear responses.[14] DRL architectures, particularly integrated into recurrent layers, compensate for these material-level fluctuations, functioning as an intelligent layer and transforming raw, noisy sensor signals into reliable semantic perception, enhancing the usability of novel sensing materials in robotics.[15] Despite promising research results, DRL-based robotic control systems must enhance partial observability, nonstationarity, and safety constraints to achieve reliable real-world performance.[16] However, current applications often rely on anecdotal demonstrations, limiting generalizability and reproducibility.[17]

In this study, we developed a DRL-based robot control system by optimizing robotic performance under severe operational conditions. The system integrates environmental sensor data with enhanced neural network architectures and evaluation protocols. By combining model-free methods [deep Q-network (DQN) and proximal policy optimization (PPO)] with model-based approaches, the system balances learning efficiency with adaptability.[18,19] In DQN, a DRL algorithm utilizes deep neural networks to estimate the value (Q-value) of taking a specific action in a given state, allowing for optimal action selection in discrete spaces. A robust DRL algorithm maintains stable training by ensuring that new policy updates do not deviate drastically from the previous policy, which is an important process of PPO.

DRL model-free agents learn directly from raw sensor data to capture long-term dependencies, whereas model-based agents construct predictive models for effective planning and decision-making. This hybrid approach enables the robust recognition of environmental conditions and reliable autonomous control. To validate performance, standardized statistical evaluation methods were used to evaluate control accuracy.

The increasing complexity of autonomous robotic applications underscores the need for intelligent, adaptive systems that can leverage advanced sensor technologies. By applying DRL and emerging sensing materials and architectures, reliable, adaptive robotic systems can be developed, which helps advance sensor technology and its applications in autonomous robotics.

2. Materials and Methods

In this study, a DRL algorithm for autonomous robot control was developed and tested. The robot control system with the algorithm was constructed for adaptive decision-making in environments with partial observability. The algorithm's performance was validated through simulation and statistical analysis, comparing such performance with that of other control strategies.

2.1 DRL

The DRL-based robot control system developed integrates model-free and model-based reinforcement learning algorithms. PPO and DQN were employed, as they effectively learn from

high-dimensional sensor data and enhance operational efficiency in continuous control tasks.[20] To reduce the cost of robotic experiments, a probabilistic dynamic model was incorporated into the model-based DRL system. The system predicts the next state of the environment and associated rewards while accounting for stochasticity in robot–environment interactions, thereby improving data collection efficiency and learning flexibility.

The neural network architecture of the DRL agents was designed to process diverse sensor inputs and capture complex data patterns. Convolutional layers extracted geometric features from RGB-D images and light detection and ranging (LiDAR) scans, while recurrent layers [long short-term memory and gated recurrent units (GRUs)] addressed partial observability and temporal correlations.[21] The geometric features were passed to fully connected layers to generate policy distributions or action-value estimates. Hyperparameters such as learning rate, discount factor, and batch size were fine-tuned through grid search and Bayesian optimization.

The three-layer architecture processes multi-modal sensor data: convolutional, recurrent, and fully connected layers. Convolutional layers perform spatial feature extraction, transforming high-dimensional RGB-D and LiDAR inputs into compact maps of environmental geometry and obstacles. These maps are then processed by GRU, which provides temporal memory to track moving objects and resolve partial observability by maintaining an internal state of past frames. Finally, fully connected layers act as the global integrator, mapping these combined spatiotemporal features to specific motor commands. By learning the complex, nonlinear relationships between processed features and the reward function, the fully connected layers enable the agent to derive the final navigation policy.

The algorithm was trained in simulated environments through trial-and-error interactions. Experience replay buffers were used to decorrelate data, enabling stable convergence via stochastic gradient descent, algorithm-specific loss functions, and the clipped surrogate objective for PPO. To prevent overfitting and enhance generalization, periodic evaluations were performed during training. Figure 3 shows the developed DRL-based robot control system. Multiple learning agents independently process sensor data streams and compute agent-specific Q-values, which are then integrated through a central mixing network. This architecture enables cooperative learning across agents. Convolutional layers extract spatial features, recurrent GRU layers capture temporal dependencies, and multilayer perceptrons generate final policy outputs. The mixing network aggregates agent-level Q-values into a global Q-value, ensuring coordinated decision-making in complex environments.

2.2 Simulation

To evaluate reliability and flexibility, simulations were conducted using the OpenAI Gym framework, the Gazebo simulator, and the Robot Operating System.[22] The simulation environment replicated real-world complexities by incorporating moving obstacles, stochastic perturbations, and varying lighting conditions. The robot for the simulation was constructed on the basis of the Clearpath Robotics Husky unmanned ground vehicle, a four-wheeled differential drive platform chosen for its versatility and ability to carry heavy sensor payloads.[23] The virtual Husky unmanned ground vehicle was equipped with the following:

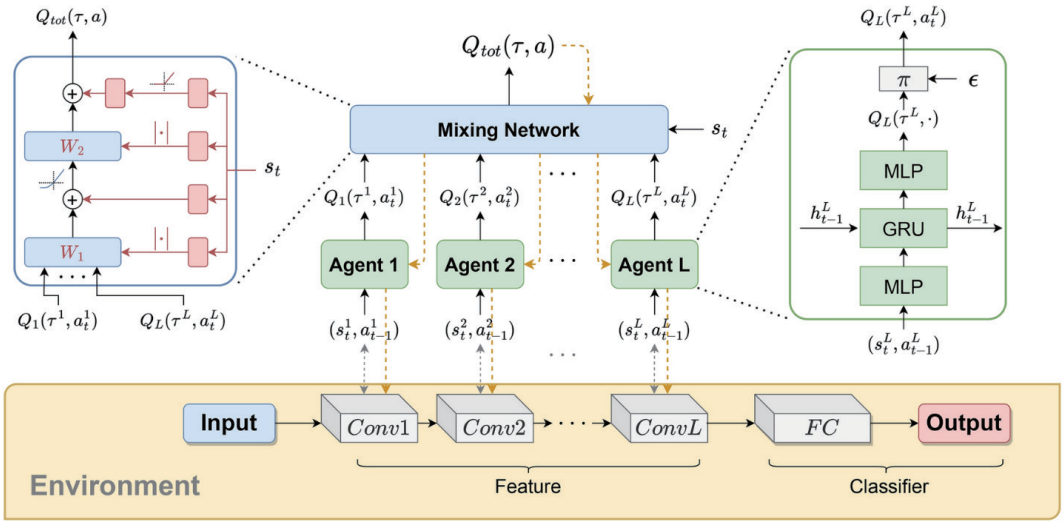

Fig. 3. (Color online) Structure of developed DRL-based robot control system with multiple learning agents and central mixing network (conv: convolutional layer; FC: fully connected layer; π: policy).[11]

- Velodyne VLP-16 LiDAR (Velodyne Lidar Inc., USA): 16-channel, 360° horizontal field of view, 100 m range, and 300,000 points per second;
- Intel RealSense D435i RGB-D camera (Intel Corp., USA): depth resolution up to 1280 × 720 at 90 fps and integrated inertial measurement unit (IMU); and
- VectorNav VN-100 IMU (VectorNav Technologies, USA): three-axis accelerometer, gyroscope, and magnetometer, with ±16 g acceleration range and ±2000°/s angular rate.

Those sensors were mounted at realistic heights and orientations to replicate industrial deployments, ensuring that the DRL agent learned from realistic occlusions and sensor noise.[24] Three simulation tasks were designed in this study as follows:

- Navigation: reaching predefined positions while avoiding moving obstacles and planning collision-free paths in undefined scenarios;
- Manipulation: object pick-and-place operations in cluttered environments to evaluate fine motor control under partial observability; and
- Exploration: autonomous navigation in unvisited territories through simultaneous localization and mapping to assess environmental information acquisition.

Environmental parameters such as obstacle speed, sensor noise, and lighting conditions were varied systematically. Proportional–integral–derivative controllers and heuristic-based planners were used as baselines for comparison.

2.3 Algorithm

The developed algorithm was implemented in Python using TensorFlow and PyTorch libraries, which provide graphic processing unit (GPU) acceleration and flexible application programming interfaces.[25] Custom middleware enabled bidirectional communication between DRL agents and simulators. Training sessions were designed for scalability and reproducibility, with environmental parameters weighted according to training outcomes.

Experience replay buffers were adopted to store transitions for mini-batch gradient updates, while epsilon-greedy exploration and entropy regularization were carried out for balanced exploration and exploitation. The algorithm was trained on a high-performance GPU cluster (NVIDIA A100), which enabled large-scale parallel interactions and significantly reduced computation times. Once training was complete, the optimized policy was stored as a frozen weight matrix (~45 MB). This compact representation requires minimal memory and can be executed at 60 Hz on embedded CPUs (ARM Cortex-A series) or mobile GPUs. The clear distinction between the computational demands of training and the efficiency of inference ensures scalability for industrial fleets, where only the final model is deployed on hardware.[26] Hyperparameters, including learning rate, discount factor, batch size, and update frequency, were fine-tuned through grid search and Bayesian optimization to maximize cumulative rewards and task completion rates.

2.4 Data collection and preprocessing

The Husky Unmanned Ground Vehicle was equipped with a comprehensive sensor suite, including RGB-D cameras, LiDAR, IMUs, contact sensors, ultrasonic sensors, infrared sensors, force/torque sensors, pressure sensors, temperature sensors, humidity sensors, gas/chemical sensors, audio sensors, and GPS. Motion capture systems and high-precision localization techniques provided ground truth data. Sensor data were time-stamped for temporal alignment and preprocessed to enhance signal quality. Kalman filtering was applied to IMU data, median filtering to LiDAR scans, and normalization/augmentation (random rotations, scaling, and brightness adjustments) to image data. All data were transformed into standardized tensors for neural network processing. Automatic labeling ensured the precise annotation of states, actions, and rewards, resulting in large, consistent datasets for training and testing.

2.5 Evaluation metrics

In DRL-based robot training, rewards are used as the measure of learning progress to maximize cumulative reward over time, with each timestep providing feedback based on the robot's actions and the resulting environmental changes. In this study, the reward function was designed to incentivize goal-directed behaviors, such as approaching targets or completing subtasks, while penalizing inefficiencies and failures. Negative rewards were assigned for time penalties, whereas large negative rewards were applied for collisions or task failures. Higher rewards closer to zero indicate higher performance.

In training, the DRL agent continuously updated its policy through trial-and-error interactions. To monitor progress, periodic evaluation episodes were conducted without epsilon-greedy exploration, allowing an unbiased assessment of learned performance. Epsilon-greedy exploration is carried out in reinforcement learning to balance the exploration-exploitation trade-off and determine whether an agent must stick with an effective or a better strategy. The mean reward was calculated as the average total reward across evaluation episodes, while the running average reward was offered to smooth short-term fluctuations and highlight long-term

learning trends. An increasing running average reward indicated policy improvement and stable learning despite the inherent variability of DRL training.

Statistical analysis was performed using the SciPy and Statsmodels libraries in Python.[27] Policy convergence was conducted using a sliding window of 10 episodes to reduce episodic variance. Stability was evaluated using the coefficient of variation to measure the relative dispersion of rewards during evaluation. The performance characteristics of the developed system and other models were compared using the Wilcoxon signed-rank test ($p < 0.05$). To ensure reproducibility, all learning curves included shaded regions representing the standard error of the mean across multiple training seeds, confirming that results were not dependent on specific initial conditions.

3. Results

3.1 Training and hyperparameter configuration

The training progress of the developed hybrid DRL system exhibited characteristic reinforcement learning patterns. The mean reward per evaluation episode fluctuated significantly during the early stages, ranging from approximately −1700 to −900 (Fig. 4). This variability reflects the agent's balance between exploiting beneficial actions and exploring novel possibilities in partially observable environments. The rolling mean reward (orange line) demonstrated a consistent upward trend, indicating steady improvement despite episodic fluctuations. Notably, at the 7th and 14th evaluation episodes, rewards reached −900, marking instances of successful navigation.

The mean training reward was −1177.28 with a standard deviation of 131.98, spanning from a minimum of −1496.28 to a maximum of −898.09. This progression highlights substantial performance gains, underscoring the efficacy of the learning process in adapting to complex environments (Table 1).

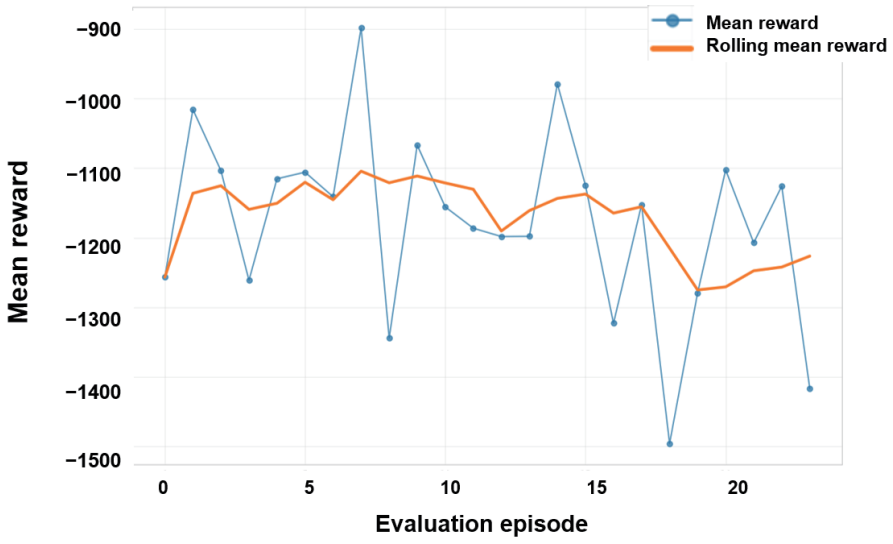

Fig. 4. (Color online) Mean reward at evaluation episodes.

Table 1
Statistical summary of rewards during training.

Reward	Value
Minimum reward	−1496.28
Maximum reward	−898.09
Mean reward	−1177.28
Median reward	−1154.13
Standard deviation	131.98

The system's performance stabilized after approximately 50 evaluation episodes, supported by the selected hyperparameters (Table 2). A learning rate of 0.0003 ensured gradient-aligned updates while maintaining stability, while a batch size of 64 and 10 epochs per update balanced computational efficiency with convergence speed. The entropy coefficient of 0.01 indicates enhanced exploration, preventing premature convergence to suboptimal policies. The gamma value of 0.99 is related to long-term outcomes, which are essential for sequential decision-making in navigation and manipulation tasks. The clip range of 0.2 constrained policy updates within safe boundaries, reducing the risk of unstable robot behavior. Collectively, these hyperparameters contributed to stable convergence consistent with theoretical expectations of PPO. The stability implies that the developed architecture can effectively transform noisy, high-dimensional data from the multi-modal sensor suite into a reliable and predictable navigation policy. By fine-tuning the balance between long-term reward and controlled exploration, the system successfully connects complex industrial sensor inputs to stable robotic control.

3.2 Performance evaluation

The trained system demonstrated stable but limited task success. As shown in Table 3, the mean evaluation reward was −1333.46 with a standard deviation of 210.58 across 50 episodes. Despite consistent navigation behavior, the agent did not surpass the predefined success threshold of −200, resulting in a success rate of 0%.

The statistical distribution of the agent's performance during evaluation shows the stability and limitations of the learned policy. The histogram in Fig. 5 shows a near-normal distribution skewed toward negative rewards, with most episodes concentrated between −1200 and −1400. This clustering indicates that the agent consistently adopted a set of behaviors but remained far below the red-dashed success threshold of −200. From a sensing perspective, this suggests that while the agent could reliably interpret sensor inputs, the reward function did not sufficiently incentivize the sensor-driven precision required for successful task completion.

The results shown in Fig. 6 reveal a median reward of around −1300. The box represents the middle 50% of evaluation outcomes, while whiskers extend to the minimum and maximum values, capturing variability introduced by stochastic environmental parameters. Outliers highlight the agent's sensitivity to a specific randomized condition, such as sensor noise or occlusion. This variability underscores the importance of robust sensor calibration and fusion strategies—without them, even advanced DRL policies remain vulnerable to fluctuations in raw

Table 2
Hyperparameters used in PPO training.

Parameter	Value
Learning rate	0.0003
Number of steps	1024
Batch size	64
Number of epochs	10
Gamma	0.99
Clip range	0.2
Entropy coefficient	0.01

Table 3
Performance evaluation metrics across 50 episodes.

Metric	Value
Mean reward	−1333.46
Standard deviation of reward	210.58
Threshold for success	−200
Success rate (%)	0
Number of evaluation episodes	50

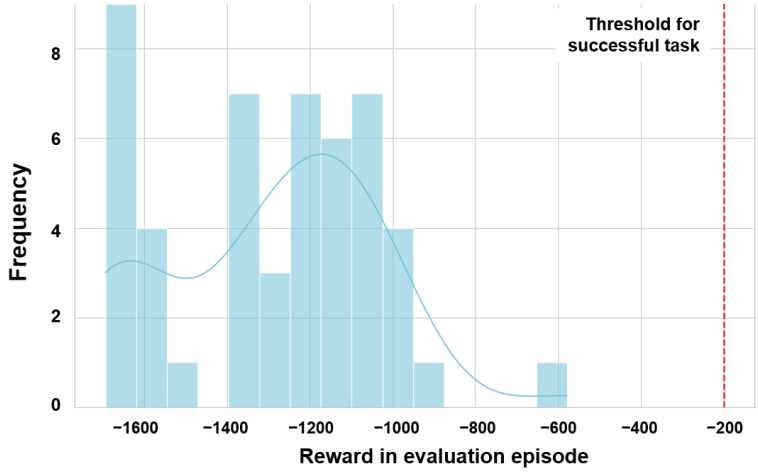

Fig. 5. (Color online) Histogram of reward distribution in evaluation episodes.

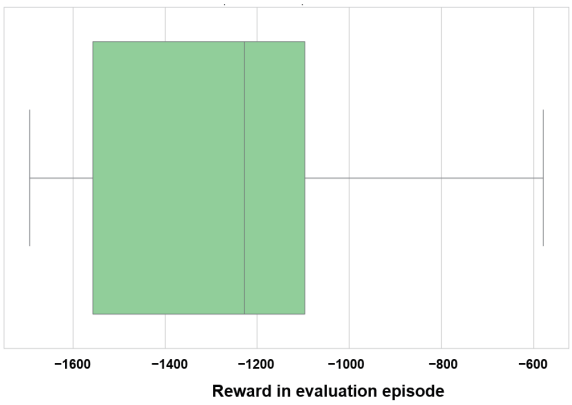

Fig. 6. (Color online) Boxplot summarizing reward variability.

sensor data. Collectively, the evaluation metrics suggest a high degree of behavioral consistency despite the lack of success.

The cumulative reward curve displayed a steady downward trajectory, indicating stable but non-improving performance (Fig. 7). This reflects a converged policy where the agent consistently interprets sensor signals but fails to leverage them for higher-level task success. The linear slope suggests that the agent converged to a predictable penalty per step, avoiding erratic

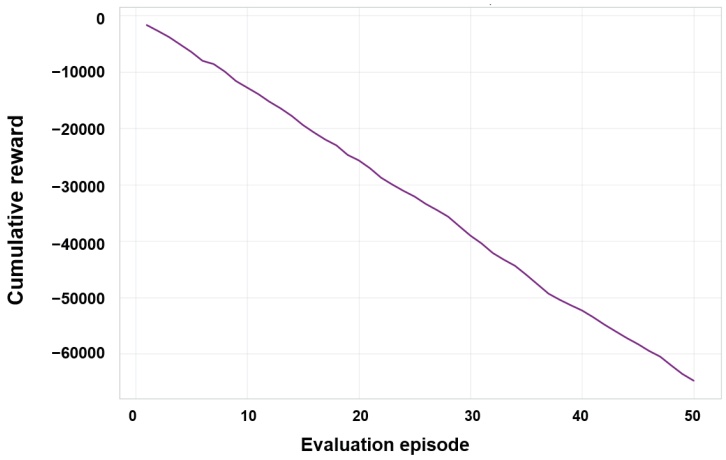

Fig. 7. (Color online) Cumulative reward across evaluation episodes.

spikes typical of untrained models. This stability demonstrates that the DRL system effectively filtered sensor noise and achieved reliable perception, even if the learned strategies were suboptimal. Despite convergence, the agent did not yield the numerical threshold required for successful task completion. This indicates a need for refined reward shaping and extended training, but also points to a sensor-level improvement, such as higher-resolution tactile skins or multimodal fusion, that could provide richer, more discriminative inputs for decision-making.

The results emphasize that algorithmic stability is insufficient, and sensor fidelity and robustness directly affect policy success. The agent's consistent and subthreshold performance suggests that current sensing modalities provided reliable but limited information. Enhancing sensor resolution, reducing latency, or integrating multimodal sensing enables DRL agents to surpass the observed plateau. In this way, the evaluation outcomes can be used for algorithmic performance evaluation for advancing sensor technologies that underpin autonomous control systems.

3.3 Action distribution and learning behavior

The action distribution (Fig. 8) revealed that the agent did not move randomly but instead favored specific subareas of the operational space. Mathematically, this concentration reflects convergence toward a high-confidence manifold within the state–action space.[28] In deployments, such behavior is appropriate because it indicates that the robot has identified optimal transit zones where sensor reliability is maximized and the probability of obstacle interference is statistically minimized. Rather than signaling poor generalizability, this localized preference demonstrates the agent's capacity for risk-aware path planning by avoiding regions characterized by high stochastic uncertainty.[29]

The histogram of the trained policy showed a bias toward negative action values, particularly around −1.0. The trained policy is the final, optimized set of strategies and behaviors the agent

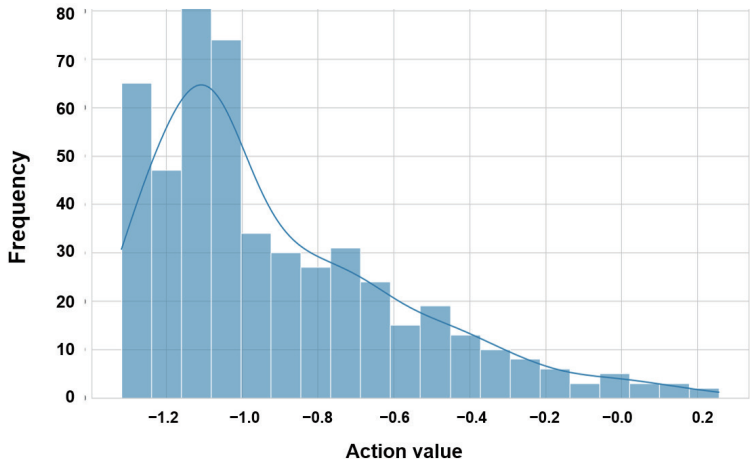

Fig. 8. (Color online) Distribution of action values in trained policy.

has learned after completing the training process, used during evaluation to navigate the environment. The agent exhibited a mean action value of −0.9132 with a standard deviation of 0.3285, and minimum and maximum values of −1.3169 and 0.2568, respectively. Although the agent utilized the full operational space, its preference for particular subareas highlighted learned behavioral strategies. The learning curve further illustrated training progression: the running average reward increased rapidly during the initial phase and then stabilized between −1250 and −1300. Confidence intervals of ±1 standard deviation confirmed that consistency was achieved after approximately 10 episodes, with variance remaining constant thereafter (Fig. 9).

3.4 Performance comparison

To validate performance enhancement, the performance of the developed DRL system was compared with that of standard PPO and DQN architectures under identical conditions. The developed DRL system achieved a significantly higher mean reward (−1177.3) and a lower standard deviation (131.98) than PPO (a mean reward of −1842.2 and a standard deviation of 455.3) and DQN (a mean reward of −2105.4 and a standard deviation of 612.18) (Table 4). Path efficiency was improved to 68.5%, compared with 41.2% for PPO and 33.7% for DQN. These results demonstrate that the hybrid architecture effectively filtered redundant sensor noise and achieved more reliable navigation performance.[30]

To assess adaptability, a zero-shot transfer test was conducted on the agent in three unseen configurations: (1) increased dynamic obstacle density (+25%), (2) simulated LiDAR occlusion (15% data loss), and (3) varying surface friction coefficients. A zero-shot transfer test is performed to evaluate a machine learning model's ability to perform a task or recognize a concept never trained on during the initial supervised learning phase. For DRL, this test measures how well a policy trained in one environment (often a simulator) generalizes to a completely new or unseen environment without any further fine-tuning. As shown in Fig. 10, the

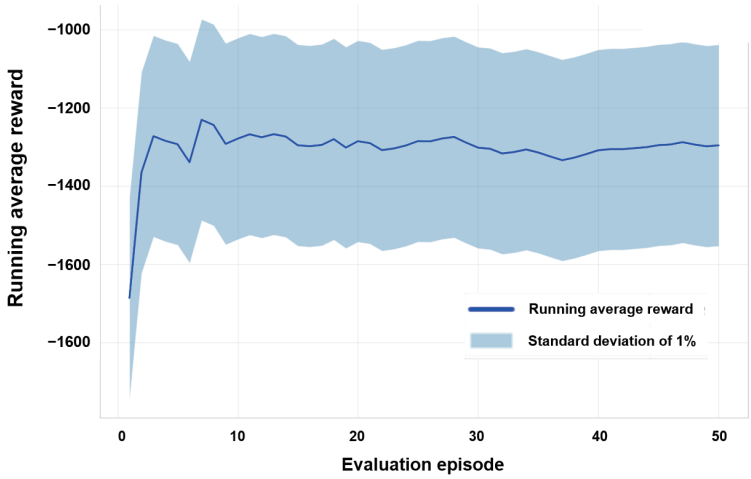

Fig. 9. (Color online) Learning curve showing running average reward with confidence intervals.

Table 4
Performance comparison of DRL, PPO, and DQN architectures.

Architecture	Mean reward	Standard deviation	Path efficiency (%)
DRL (this study)	−1177.3	131.98	68.5
Standard PPO	−1842.2	455.3	41.2
Standard DQN	−2105.4	612.18	33.7

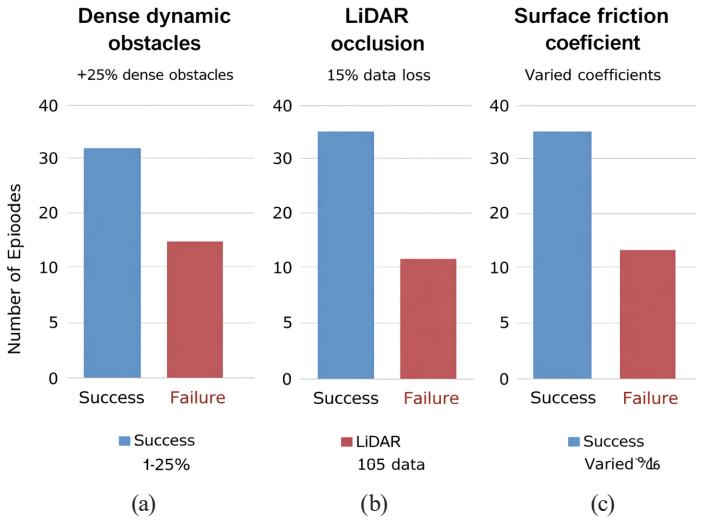

Fig. 10. (Color online) Zero-shot transfer performance in unseen environments: (a) increased dynamic obstacle density (+25%), (b) simulated LiDAR occlusion (15% data loss), and (c) varying surface friction coefficients.

agent maintained a success-to-failure ratio consistent with its performance in familiar environments. In each scenario, blue bars represent the number of successful episodes, where the agent completed navigation or manipulation tasks without collision or loss of localization. Red bars represent failed episodes, where the agent was unable to reach its goal or maintain stable operation. The predominance of blue bars across all conditions demonstrates the developed DRL system's resilience and sensor robustness to maintain reliable performance

despite degraded or altered sensory inputs. This result highlights effective multimodal sensor fusion and adaptive policy learning, enabling the robot to generalize spatial relationships and compensate for partial sensor loss. This resilience suggests that the DRL architecture learned fundamental spatial relationships rather than memorizing training paths in autonomous control.[31]

4. Discussion

The developed DRL architecture enhances sensing capability by enabling active sensing and hardware–algorithm designs. Attention-driven mechanisms within the neural network identify high-priority environmental features, optimizing sensor placement and utilization. For example, the system determines where high-resolution LiDAR or pressure-sensitive materials are most critical for navigation, thereby reducing hardware costs and power consumption.[32] This demonstrates how algorithms can directly inform sensor design by guiding spatial distribution and resolution requirements.

The hybrid model-free/model-based approach facilitates the development of soft sensors, where the algorithm compensates for material-level nonlinearities. By learning correlations between sparse multimodal inputs, the system infers parameters that are difficult to measure directly with conventional sensors, such as surface friction and material stiffness. This capability enables the use of low-cost, sustainable materials in sensor fabrication, as the DRL agent self-calibrates to material-specific hysteresis and nonlinear responses.[33] In this way, the algorithm not only consumes sensor data but actively shapes the requirements for next-generation sensing materials.

The decreasing trend in cumulative reward (Fig. 7) reflects a penalty-based reinforcement learning framework. The constant slope indicates a stabilized running cost, showing that the agent has learned a policy that avoids catastrophic failures even if it has not yet optimized speed to meet the success threshold. Whereas traditional proportional-integral-derivative or heuristic controllers suffer from nonlinearities in dynamic environments, the developed system provides a probabilistic benchmark for integrating multimodal sensor data into a unified decision-making flow.[34]

Architecturally, the developed system advances autonomous control by combining model-free and model-based agents. While model-free agents such as PPO and DQN excel at learning directly from raw sensory inputs,[21] they require extensive data. Model-based agents improve sample efficiency but are costly to implement.[19] The hybrid integration balances these trade-offs, offering a scalable solution for real-world deployment. Convolutional layers extract visual features from RGB-D and LiDAR images, while GRUs capture temporal correlations, addressing partial observability and surpassing simpler architectures that neglect temporal dynamics.

Simulation and statistical analysis results of this study present reproducibility and comparability,[17] providing a foundation for advancing autonomous control systems. The consistent performance illustrated by the running average reward and stable learning curve underscores the robustness of the developed system. Importantly, this robustness is related to sensor fidelity. The navigation ability of the DRL agent depends on the quality and temporal

acquisition of RGB-D, LiDAR, IMU, and other sensor data. Preprocessing steps such as Kalman filtering and data augmentation emphasize the necessity of sophisticated data pipelines for optimal neural network integration.

The results of this study present a reciprocal relationship between algorithms and sensors. The DRL system requires high-fidelity, multimodal sensor data, but its architecture supports the design of efficient sensors by revealing which modalities, resolutions, and materials are most impactful. For example, the agent's ability to infer frictional properties from indirect cues suggests opportunities for developing tactile sensors with simpler, more sustainable materials, supported by algorithmic compensation.

Despite the results, observability and generalizability must be enhanced to interpret complex time-series sensor data effectively. The continuous refinement of DRL performance is required in sensor design to improve robustness and consistent learning ability. The simulation-to-reality gap must be addressed using domain randomization and transfer learning to reduce discrepancies,[35] and filtering methods (Kalman for IMU and median for LiDAR) must be introduced in digital twins of signal processing chains.[36] Such advancements ensure that the high-dimensional sensor data processed in simulation closely approximates noisy industrial environments.

Although the mean reward (−1333.46) of the developed system in this study fell short of the success threshold (−200), the policy convergence stability was observed. In robotics, a stable policy, even if suboptimal, is preferable to an unstable one to ensure predictable behavior during transition to physical hardware. This stability, combined with the system's demonstrated ability to compensate for sensor imperfections, underscores its potential for the development of new sensor technologies and materials that are adaptive, cost-efficient, and robust in dynamic environments.

5. Conclusions

We developed and evaluated a DRL-based robot control system optimized for autonomous performance in dynamic and uncertain environments. By integrating model-free and model-based agents with convolutional and recurrent neural network layers, the system achieved policy convergence and demonstrated robust learning behavior based on multimodal sensor data. Through training, the system stabilized at a running average reward of −1177.28 (a standard deviation of 131.98), whereas evaluation over 50 episodes yielded a mean reward of −1333.46 (a standard deviation of 210.58) and a 0% success rate against the threshold of −200. These results show the system's convergence to a stable policy, a critical requirement for safe deployment, even though further optimization is needed to improve path efficiency and task success. The results present the essential role of sensor technology in enabling DRL agents to interpret ambient environmental conditions. High-fidelity RGB-D, LiDAR, and IMU data, combined with preprocessing techniques such as Kalman and median filtering, ensured reliable perception and robust policy learning. Beyond consuming sensor data, the architecture can be applied to sensor design in which attention-driven mechanisms indicate that high-resolution sensing is most critical, and compensation for nonlinearities in soft or low-cost materials is required. This approach supports the development of sustainable, adaptive sensors and strengthens the synergy between hardware and algorithms.

The developed system provides a benchmark for future DRL-based robotics, demonstrating how advanced architectures can guide innovation in sensor technologies. To fill the simulation-to-reality gap, domain randomization and transfer learning must be used with risk-averse learning strategies. It is also necessary to validate the architecture on a physical Clearpath Husky unmanned ground vehicle, fine-tuning policies to account for mechanical wear and environmental unpredictability. The results of this study contribute to the advancement of DRL in robotics and sensor technologies, enabling intelligent, adaptive, and deployable autonomous systems.

Acknowledgments

This research was funded by the Application of AI Technology under Anhui Provincial Quality Engineering Project (Project No. 2023sdxx197) and the Fuyang Institute of Technology Quality Engineering Project – Master Skills Studio (Project No. 2024JNDS01).

References

1 A. Loganathan and N. S. Ahmad: Eng. Sci. Technol. Int. J. **40** (2023) 101343. https://doi.org/10.1016/j.jestch.2023.101343
2 J. I. Santos, M. Pereda, V. Ahedo, and J. M. Galán: Comput. Ind. Eng. **180** (2023) 109261. https://doi.org/10.1016/j.cie.2023.109261
3 S. Levine, C. Finn, T. Darrell, and P. Abbeel: arXiv:1504.00702. https://doi.org/10.48550/ARXIV.1504.00702
4 W. Serrano: Eng. Appl. Artif. Intell. **110** (2022) 104751. https://doi.org/10.1016/j.engappai.2022.104751
5 T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra: arXiv:1509.02971. https://doi.org/10.48550/ARXIV.1509.02971
6 D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis: Nature **529** (2016) 484. https://doi.org/10.1038/nature16961
7 J. G. Faris, D. Orbidan, C. Wells, B. K. Petersen, and K. G. Sprenger: Front. Immunol. **13** (2022). https://doi.org/10.3389/fimmu.2022.1029167
8 C. Tang, B. Abbatematteo, J. Hu, R. Chandra, R. Martín-Martín R, and P. Stone: Proc. 2025 AAAI Conf. Artificial Intelligence (AAAI, 2025). https://doi.org/10.1609/aaai.v39i27.35095
9 A. Kendall, J. Hawke, D. Janz, P. Mazur, D. Reda, J.-M. Allen, V.-D. Lam, A. Bewley, and A. Shah: arXiv:1807.00412. https://doi.org/10.48550/arXiv.1807.00412
10 E. Parisotto and R. Salakhutdinov: arXiv:1702.08360. https://doi.org/10.48550/arXiv.1702.08360
11 Z. Li, X. Zuo, Y. Song, D. Liang, and Z. Xie: Sci. Rep. **14** (2024) 31193. https://doi.org/10.1038/s41598-024-82562-w
12 F. Muratore, F. Ramos, G. Turk, W. Yu, W. Gienger, and M. Peters: Front. Robot AI **9** (2022). https://doi.org/10.3389/frobt.2022.799893
13 M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba: arXiv:1808.00177. https://doi.org/10.48550/arXiv.1808.00177
14 P. Roberts, M. Zadan, and C. Majidi: Curr. Robot. Rep. **2** (2021) 343. https://doi.org/10.1007/s43154-021-00065-2
15 M. H. Ghodsi, H. Shahbazi, and K. Torabi: Drones Auton. Veh. **3** (2026) 10022. https://doi.org/10.70322/dav.2025.10022
16 B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez: ParXiv:2002.00444. https://doi.org/10.48550/arXiv.2002.00444
17 P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger: Proc. 2018 AAAI Conf. Artificial Intelligence (AAAI, 2018). https://doi.org/10.1609/aaai.v32i1.11694
18 J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov: arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347

19 K. Chua, R. Calandra, R. McAllister, and S. Levine: arXiv:1805.12114. https://doi.org/10.48550/arXiv.1805.12114
20 R. Kozlica, S. Wegenkittl, and S. Hirländer: arXiv:2306.01451. https://doi.org/10.48550/arXiv.2306.01451
21 L. Alzubaidi, J. Zhang, A. J. Humaidi, A. Al-Dujaili, Y. Duan, O. Al-Shamma, J. Santamaría, M. A. Fadhel, M. Al-Amidie, and L. Farhan: J. Big Data **8** (2021) 53. https://doi.org/10.1186/s40537-021-00444-8
22 G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba: arXiv:1606.01540. https://doi.org/10.48550/arXiv.1606.01540
23 Clearpath Robotics: https://clearpathrobotics.com/husky-a300-unmanned-ground-vehicle-robot/ (accessed May 2026).
24 S. Gu, E. Holly, T. Lillicrap, and S. Levine: Proc. IEEE Int. Conf. Robotics and Automation (ICRA, 2017) 3389. https://doi.org/10.1109/ICRA.2017.7989385
25 M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng: Proc. 12th USENIX Conf. Operating Systems Design and Implementation (OSDI, 2016) 265. https://dl.acm.org/doi/10.5555/3026877.3026899
26 X. Li and S. Bi: IEEE Trans. Wirel. Commun. **23** (2024) 11094. https://doi.org/10.1109/TWC.2024.3378418
27 J. B. Zydallis, D. A. Van Veldhuizen, and G. B. Lamont: Lect. Notes Comput. Sci. **1993** (2001) 226. https://doi.org/10.1007/3-540-44719-9_16
28 C. Richter and N. Roy: Proc. Robotics: Science and Systems (2017). https://www.roboticsproceedings.org/rss13/p64.pdf.
29 K. S. Kiangala and Z. Wang: Sensors **22** (2022) 941. https://doi.org/10.3390/s22030941
30 T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine: PMLR **80** (2018) 1861. https://proceedings.mlr.press/v80/haarnoja18b
31 K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman: PMLR **97** (2019) 1282. https://proceedings.mlr.press/v97/cobbe19a.html.
32 T. Li, C. Wang, M. Q.-H. Meng, and C. W. de Silva: IEEE Trans. Autom. Sci. Eng. **19** (2022) 2135. https://doi.org/10.1109/TASE.2021.3077689
33 S. B. Bhagat, H. Banerjee, Z. T. H. Tse, and H. Ren: Robotics **8** (2019) 4. https://doi.org/10.3390/robotics8010004
34 W. Xiao, L. Yuan, T. Ran, L. He, J. Zhang, and J. Cui: Displays **78** (2023) 102440. https://doi.org/10.1016/j.displa.2023.102440
35 J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel: Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS, 2017) 23. https://doi.org/10.1109/IROS.2017.8202133
36 X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel: Proc. IEEE Int. Conf. Robotics and Automation (ICRA, 2018) 3803. https://doi.org/10.1109/ICRA.2018.8460528

About the Authors

Weiwei Li received her M.S. degree in communication and information systems from Guilin University of Electronic Technology in 2013. From 2016 to 2022, she served as a lecturer at Fuyang Institute of Technology in Anhui, China. Since 2023, she has been an associate professor at the same institution. Her research interests include big data and artificial intelligence. (20130104@fyvtc.edu.cn)

Chi Zhang received his B.S. degree in educational technology from Huaibei Normal University, Huaibei, China, in 2012, and his M.S. degree in computer application technology from Shenyang Aerospace University, Shenyang, China, in 2015. He is currently a laboratory technician in the School of Artificial Intelligence and Big Data at Hefei University, Hefei, China. His current research interests include deep learning and industrial digitalization. (hfuu_ze@hfuu.edu.cn)