***A Comparative Study of DQN and DDPG for Continuous Control in Reinforcement Learning***

*By: Abarna Rajarethinam*

ABSTRACT

In this paper, I explore the limitations of Deep Q-Networks (DQN) in handling continuous action spaces and evaluate the effectiveness of Deep Deterministic Policy Gradient (DDPG) as an alternative. Although DQN is widely effective in environments with discrete action spaces, its application to continuous control tasks such as Pendulum-v0 requires discretizing the action space into a finite set of values. This limitation often hinders the agent’s ability to learn fine-grained control, resulting in suboptimal performance. In contrast, DDPG is designed to operate in continuous domains through a model-free, actor-critic framework that enables more precise and flexible action selection. Through a comparative experiment using consistent training setups, I observed that DDPG was able to learn smoother and more stable policies, outperforming DQN in both learning efficiency and final performance. This comparison highlights the practical importance of selecting reinforcement learning algorithms based on the structure of the action space and supports the adoption of policy-based methods for real-world continuous control problems.

INTRODUCTION

Reinforcement Learning (RL) has emerged as a powerful framework for enabling agents to learn optimal behaviours through interaction with dynamic environments. Unlike supervised learning, where models are trained on labelled data, RL relies on feedback in the form of rewards to optimise learning. This makes it well-suited for sequential decision-making problems where the outcome of an action may only be evident after several steps. RL has found applications across a broad range of domains, including robotics, game playing, autonomous navigation, and recommendation systems.

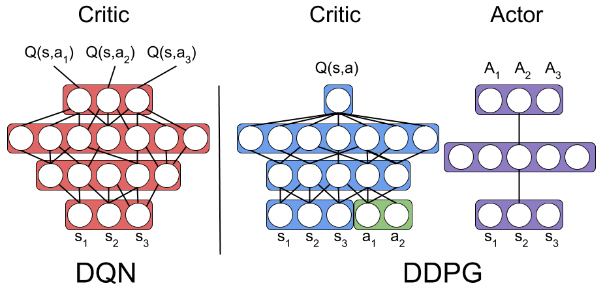
At the core of RL lies the agent–environment loop, where the agent observes the current state, selects an action, receives a scalar reward, and transitions to a new state. Over time, the agent’s objective is to learn a policy that maximises the cumulative reward. The success of RL in high-dimensional settings has been amplified by combining it with deep learning, giving rise to deep reinforcement learning (DRL), where neural networks are used to approximate policies and value functions.

One of the most notable DRL algorithms is the Deep Q-Network (DQN), which demonstrated human-level performance on a wide array of Atari video games using only raw pixel input (Mnih et al., 2015). In these tasks, DQN uses a convolutional neural network to estimate the action-value (Q) function, allowing the agent to learn effective policies in discrete action spaces. DQN’s success was largely enabled by innovations such as experience replay and the use of target networks, which stabilised training with deep function approximators.

Despite these successes, DQN is fundamentally limited to discrete action domains. In continuous control tasks, such as robotic arm manipulation or torque-based systems like Pendulum-v0, DQN cannot directly operate on real-valued action spaces. Applying DQN in such settings requires discretising the action space into a fixed set of bins, which introduces several drawbacks. The loss of precision hampers the agent’s ability to learn smooth control policies, and the number of possible actions can grow exponentially with task complexity, making training inefficient and unstable.

To address this limitation, I examine the Deep Deterministic Policy Gradient (DDPG) algorithm (Lillicrap et al., 2015), which is specifically designed for continuous action spaces. DDPG is a model-free, off-policy actor-critic method that combines the strengths of deterministic policy gradients with stabilising techniques from DQN. Unlike DQN, the actor network in DDPG outputs continuous action values directly, allowing the agent to learn fine-grained control strategies without discretisation.

In this work, I compare DQN and DDPG using the Pendulum-v0 environment, a standard benchmark in continuous control. This environment requires the agent to apply continuous torque to swing and balance a pendulum in the upright position. Through this comparison, I aim to analyse the architectural differences between the two algorithms, evaluate their performance in terms of reward and stability, and demonstrate the advantages of actor-critic methods like DDPG in domains that demand continuous, real-time control. Figure 1 illustrates the core architectural differences between DQN and DDPG, highlighting how the actor-critic structure in DDPG enables continuous control.



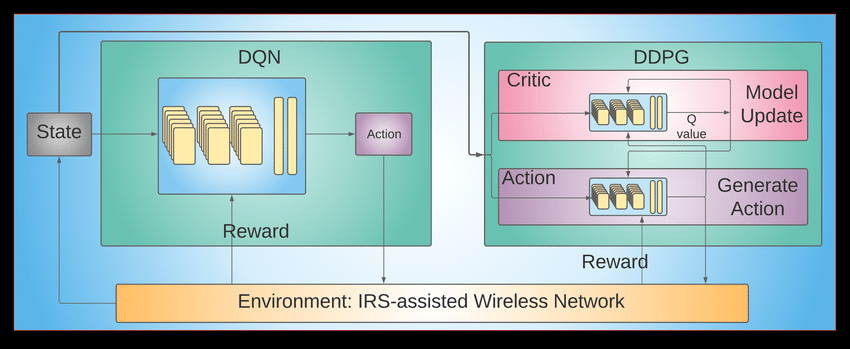
**Figure 1:** Architectural comparison between DQN, DDPG, and Policy Gradient methods. DQN relies on a single Q-network to estimate action-values for discrete actions. In contrast, DDPG introduces separate actor and critic networks, enabling continuous action output and improved learning in continuous control tasks like Pendulum-v0.

RELATED WORKS

Both DQN and DDPG have played pivotal roles in advancing deep reinforcement learning, particularly in their ability to address different types of action spaces. This section reviews the key developments of each algorithm, with a focus on their architectural design and how that impacts their suitability for continuous control.

The Deep Q-Network (DQN) algorithm, introduced by Mnih et al. (2015), was the first to successfully integrate Q-learning with deep neural networks. It uses a single neural network that maps input states to a vector of Q-values, each corresponding to a discrete action. This structure allows the agent to select the action with the highest predicted Q-value using an argmax operation. To stabilise learning, DQN employs experience replay and a separate target network that periodically updates its weights from the main network. However, the reliance on a discrete action output means that DQN is not directly compatible with continuous action spaces. Applying it to such environments requires discretisation of the action space, which increases action dimensionality and introduces approximation errors.

In contrast, the Deep Deterministic Policy Gradient (DDPG) algorithm, proposed by Lillicrap et al. (2015), employs a dual-network actor-critic architecture that is inherently designed for continuous action spaces. The **actor network** maps input states directly to continuous actions, producing deterministic outputs rather than probabilities or Q-values. The **critic network**, on the other hand, takes both the current state and the action (either from the actor or sampled) as input and estimates the corresponding Q-value. This separation of concerns allows DDPG to decouple action selection from action evaluation, enabling more flexible policy updates.



**Figure 2:** High-level architectural comparison between DQN and DDPG. DQN uses a single Q-network to map states to action values, selecting actions via argmax. DDPG introduces separate actor and critic networks: the actor generates continuous actions directly, while the critic evaluates them to compute Q-values for learning updates. This separation enables more precise and stable control in continuous environments. \* *Although this figure originates from a wireless network application, the architectural structure of DQN and DDPG illustrated here remains consistent across most reinforcement learning environments. \**

DDPG also approaches exploration differently from DQN. In class, we learned how DQN explores by randomly selecting actions using an ε-greedy strategy, which gradually becomes more focused on high-value actions over time. While this works in simple environments with clear action choices, it becomes limiting in tasks that require smooth, precise control. DDPG addresses this by adding a small amount of variation to the actions generated by its actor network, allowing the agent to try slightly different actions from what it has already learned. This helps the agent explore more effectively in environments like Pendulum, where the ideal action is not just “left” or “right,” but something in between.

In my implementation of DDPG, I designed the actor network to output continuous real-valued actions within the allowed range of the environment (e.g., -2 to 2 in Pendulum-v0). Instead of relying on discrete action selection, the actor generates smooth and precise actions directly. To enable exploration, I added Gaussian noise to the actor's output during training, allowing the agent to try nearby actions and improve gradually. This approach allowed me to fully utilize DDPG’s strength in handling continuous control tasks without needing to discretize the action space.

Although DQN and DDPG both use methods like replay buffers and target networks to improve learning stability their internal designs are quite different. DQN uses a single network to evaluate actions, while DDPG separates the process into two roles: one for choosing actions (actor) and one for judging them (critic). This allows DDPG to handle continuous action spaces more naturally, making it better suited for real-world control tasks like robotics. The next section presents a side-by-side comparison of both methods and describes how they were applied in this experiment.

EXPERIMENT

To compare the effectiveness of Deep Q-Network (DQN) and Deep Deterministic Policy Gradient (DDPG), I implemented both algorithms using the classic Pendulum-v0 environment from OpenAI Gym. This environment provides a well-known benchmark for continuous control tasks, where the goal is to apply torque to a pendulum to swing it upright and keep it balanced. The observation space is continuous consisting of the pendulum’s angular position and velocity, while the action space is also continuous, representing the applied torque.

One key aspect of reinforcement learning (RL) that makes it different from supervised learning is that no dataset is required. Instead, the agent learns by interacting directly with the environment, generating data on the fly. As such, this experiment did not require any preprocessing or external datasets.

Because DQN is designed for discrete action spaces, I had to manually discretize the continuous action space of the Pendulum environment into five fixed torque levels. While this allowed DQN to function, it came at the cost of coarse control—the agent could not fine-tune its actions, which limited its ability to learn smooth balancing behaviours. This limitation became more apparent as training progressed.

On the other hand, DDPG’s actor-critic architecture allowed the agent to output continuous torque values directly. This meant DDPG did not require any discretization and could learn much more precise and stable control policies. I observed that this advantage led to smoother learning curves and better final performance.

To ensure fair comparison and reproducibility, I set random seeds for NumPy, TensorFlow, and Python's random module. I trained both agents for 100 episodes each and tracked their total rewards per episode.

DISCUSSION

The experimental results with Pendulum-v0 clearly illustrated the practical consequences of using DQN versus DDPG in a continuous control task. While both algorithms successfully learned to improve performance over time, their architectural differences led to distinctly different learning behaviours and outcomes.

DQN, being rooted in value-based Q-learning, is relatively simple to implement and often serves as a good starting point for understanding reinforcement learning. However, the need to discretize the action space in a continuous environment like Pendulum introduced significant limitations. The agent’s inability to produce fine-grained actions resulted in jerky, less stable movements and slower convergence. This reflects a broader issue: when discretization is too coarse, control is imprecise; when too fine, the action space explodes, harming sample efficiency and training speed.

In contrast, DDPG’s actor-critic architecture allowed for direct output of continuous actions, which translated to smoother torque application and faster adaptation to the environment's dynamics. This difference was evident in the more stable reward progression and higher performance achieved by DDPG. As shown in **Figure 3**, the DDPG agent consistently outperformed the DQN agent in terms of cumulative reward, highlighting its advantage in continuous domains.

However, the added flexibility of DDPG comes with trade-offs. The algorithm required more components, an actor and critic network, each with target counterparts, as well as careful tuning of hyperparameters and stabilizing techniques like soft updates and noise injection. These complexities increase both the computational overhead and the difficulty of achieving robust performance without overfitting or divergence.

Ultimately, this experiment demonstrates that while DQN is simpler and more intuitive, it struggles in tasks that demand smooth, precise actions. DDPG, though more sensitive and resource-intensive, offers clear advantages in continuous domains where such control fidelity is essential.

***Table 1****: Summary of Differences between DQN and DDPG*

| **Feature** | **DQN** | **DDPG** |
| --- | --- | --- |
| Algorithm Type | Value-based (Q-learning) | Actor-Critic |
| Action Space | Requires discretization | Continuous outputs directly |
| Network Architecture | Single Q-network (+ target) | Actor + Critic networks (+ targets) |
| Exploration Strategy | ε-greedy | Ornstein-Uhlenbeck noise |
| Sample Efficiency | Lower in continuous settings | Higher due to continuous actions |
| Stability Requirements | Moderate | High (requires soft updates, tuning, etc.) |
| Implementation Complexity | Simple | More complex |
| Real-World Suitability | Limited in robotics | Suitable for robotics and control systems |

A graph of blue and orange lines

AI-generated content may be incorrect.

**Figure 3**: Reward comparison between DQN and DDPG agents over 100 episodes in the Pendulum-v0 environment.

CONCLUSION

In this study, I set out to compare Deep Q-Networks (DQN) and Deep Deterministic Policy Gradient (DDPG) in the context of continuous control using the Pendulum-v0 environment. Through hands-on implementation and experimentation, I observed key differences in how each algorithm handles action representation and policy learning.

DQN, while easier to understand and implement, struggled with the limitations imposed by action discretization. The coarse action space led to jerky movements and suboptimal control, which was reflected in the lower reward values across episodes. In contrast, DDPG, with its actor-critic architecture and continuous action output, was able to learn smoother and more refined control strategies. Although DDPG required more careful tuning and additional components such as target networks and noise processes, it ultimately yielded superior performance.

This experiment reinforced a core idea in reinforcement learning: the choice of algorithm should be guided by the nature of the environment. For tasks involving continuous, real-time control, like robotics or mechanical systems, policy gradient methods such as DDPG are far more appropriate than value-based methods like DQN. While DQN remains a strong baseline for discrete problems, this project demonstrated how architectural flexibility and direct-action modelling give actor-critic methods a significant edge in continuous domains.

REFERENCES

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). *Continuous control with deep reinforcement learning*. arXiv preprint arXiv:1509.02971. <https://arxiv.org/abs/1509.02971>

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Hassabis, D. (2015). *Human-level control through deep reinforcement learning*. Nature, 518(7540), 529–533. <https://doi.org/10.1038/nature14236>

Shah, K. (2020, July 28). *Deep Deterministic Policy Gradient (DDPG): An Off-Policy Reinforcement Learning Algorithm*. Medium. Retrieved from <https://medium.com/intro-to-artificial-intelligence/deep-deterministic-policy-gradient-ddpg-an-off-policy-reinforcement-learning-algorithm-38ca8698131b>

Zhou, Z., Chen, X., Zhao, W., & Bennis, M. (2020). *Learning to schedule real-time traffic in intelligent reflecting surface-assisted wireless networks*. ResearchGate. <https://www.researchgate.net/figure/n-RL-the-DQN-and-DDPG-architectures-accept-same-state-channel-data-and-received-SNR_fig6_344159696>