Investigating the Impact of Architectural Modularity and Design Choices on the Robustness of Deep Reinforcement Learning in Noisy Environments

Tinaabishegan Baladewan

I. Introduction

Deep Reinforcement Learning (RL) has achieved remarkable success in a range of complex decision-making tasks, including robotic control, and autonomous navigation [4], [5]. Nevertheless, RL agents often exhibit substantial sensitivity to environmental and operational noise [2]. Such noise manifesting through uncalibrated sensors, fluctuating environmental conditions, and intermittent actuator failures can significantly degrade learning stability and overall performance. In real-world applications, where the complexity and unpredictability of the environment are higher, the robustness of RL agents against various noise sources becomes a critical factor for their practical deployment. Increasing the robustness of RL agents necessitates strategies that move beyond mere denoising or after-the-fact compensation. While denoising mechanisms can improve signal quality, they may incur substantial computational and implementation costs, and do not fully address core architectural vulnerabilities. Instead, a promising avenue lies in exploring how carefully designed architectural components can intrinsically enhance agents' ability to learn and retain optimal policies under noisy and uncertain conditions. This is particularly important as real-world systems often encounter diverse, non-stationary noise, and failing to manage these disturbances can lead to safety risks.

A. Motivation and Problem Statement

Previous work shows that adjusting RL architectures and hyperparameters can influence resilience to noise [2]. Yet, we lack clarity on how integrating Double Q-learning, PER, Dueling, NoisyNets, and NROWAN affects performance under varied noise in discrete-action tasks. Modularity in RL [1], [3] can improve efficiency, but studies mainly cover deterministic or simple noisy setups. Similarly, noise-aware RL [2] often assumes continuous or isolated noise. We lack understanding of whether modular architectures can handle multiple, transient noise sources, especially when noise exposure is limited and irregular. While reducing noise scales can improve learning performance [2], there is a paucity of studies investigating how architectural choices (e.g., Double DQN, PER, Dueling, NoisyNets, NROWAN) interact with transient and mixed noise sources. Existing work on modularity [1], [3] and noise reduction [2] lacks a comprehensive approach to integrating these insights. Although modular RL agents show promise in simpler scenarios [1], [3], their performance under intermittent, intense noise is unclear. Real environments often present noise sporadically. Can agents learn from limited, intense noise episodes and retain those strategies afterward? Most existing research assumes continuous or long-duration exposure to noise. In contrast, real environments may present noise sporadically. How well do different architectures learn from limited but intense noise episodes, and can they "remember" how to handle such noise once it ceases to be present?

B. Research Questions

1) How do different DQN-based architectural enhancements collectively affect learning performance and robustness under environmental, sensor, and action noise, especially when noise exposure is limited to early training episodes?

2) To what extent does a hierarchically modular architecture, improve resilience and adaptability to varied noise types, compared to non-modular or single-level architectures?

These questions aim to bridge the gaps, exploring whether architectures that incorporate both modularity and advanced design choices provide better handling of complex, transient noise.

C. Contributions

In this study, we propose a comprehensive experimental framework to evaluate multiple DQN-based architectures. Our key contributions include:

1) Combined Noise Injection

We introduce a training protocol that applies environmental wind, sensor Gaussian noise, and motor/engine failure rates at the start of training. These noises are sustained for a certain number of episodes and then gradually annealed to zero, simulating limited and transient real-world noise exposure.

2) Comparative Architectural Evaluation

We systematically compare standard DQN, DoubleDQN-PER-Dueling, NoisyNets, and NROWAN variants, alongside hierarchical RL architectures, to assess which combinations yield robust performance under various noises.

3) Realistic Training Conditions

By limiting noise exposure to early training stages, we assess architectures' ability to "learn

from limited chances" to handle noise and subsequently maintain stable performance when noise levels return to baseline. This provides insights into how architectures can internalise robustness.

II. Background

Reinforcement Learning (RL) enables agents to learn optimal policies by maximising cumulative rewards through trial-and-error interactions with an environment [4]. Deep Q-Networks (DQN) [5] introduced neural function approximators for value estimation in discrete action spaces, achieving human-level performance on Atari benchmarks. Despite these successes, DQNs are sensitive to noisy conditions and often struggle to maintain stable policies. Fortunato et al. [1] proposed NoisyNets to promote exploration by injecting parametric noise into neural network weights, allowing agents to adaptively balance exploration and exploitation. This makes them particularly promising for environments with action noise, where exploration under uncertain conditions is crucial. However, the effectiveness of NoisyNets in episodic or transient noise scenarios has not been fully explored.

Similarly, NROWAN [6] introduces auxiliary networks to enhance diversity in function approximation, improving stability and exploration. Its ability to handle complex noise patterns and its synergy with modular and hierarchical architectures remains underexplored, especially when noise is intermittent or occurs in bursts.

Recent studies [1], [3] highlight the potential of modular and hierarchical RL architectures to facilitate more efficient representation learning, particularly under deterministic or simple noisy conditions. These designs can help decompose tasks and handle competing objectives, potentially improving resilience to noise. However, their effectiveness under complex and transient noise conditions—reflecting real-world scenarios—has yet to be systematically evaluated.

Taken together, these works underscore the potential benefits of advanced architectural modifications and modularity, particularly for handling action noise and transient uncertainties. Our study builds on these foundations, systematically combining these approaches and evaluating their robustness in realistic noisy environments.

III. Methods

This investigation evaluates how different RL architectures particularly DQN variants perform under conditions of limited and transient noise exposure. We consider environmental, sensor, and action-level noise, each present only during the early training phase and then gradually removed. Our experiments aim to identify architectures that inherently develop robust strategies, enabling stable long-term performance even after noise subsides.

A. Experimental Design

We conduct experiments in the OpenAI Gym [7] environments CartPole-v1 and LunarLander-v3. Each environment presents distinct challenges:

- CartPole-v1: A simple environment where the agent must balance a pole atop a cart by choosing left or right actions. The state space includes position, velocity, angle, and angular velocity. The agent's goal is to keep the pole upright.
- LunarLander-v3: A more complex environment where the agent controls a lunar module's thrusters to land smoothly on a landing pad. The observation space includes vertical/horizontal position, velocity, angle, and leg contacts, while the action space involves discrete engine firing patterns.

For each environment, the agent trains for a specified number of episodes. During training, we introduce combined noise at the start:

- Environmental Noise (Wind): Applied as a horizontal force, initially set at 0.2.
- Sensor Noise (Gaussian): Added to state observations with a standard deviation of 0.2.
- Action Noise (Engine Failure): A 30% chance that intended actions fail, simulating motor faults.

These noise scales remain constant for an initial period (25 episodes for CartPole-v1, 150 episodes for LunarLander-v3) and then anneal linearly to 0 over a set duration (up to 125 episodes for CartPole-v1 and 350 episodes for LunarLander-v3). After the noise returns to zero, the agent continues training in a noise-free setting, testing how well it has internalised strategies to handle uncertain conditions.

Post-training, each trained model is validated by running 500 episodes under a fixed moderate noise level. This validation tests the architecture's ability to recall and robustly handle noise after limited noisy exposure during training.

B. Architectures and Components

This study evaluates several DQN-based architectures, each designed to address specific challenges in reinforcement learning under noisy conditions:

DQN [5]: The baseline architecture that utilises a standard deep neural network to approximate the Q-function. It serves as a benchmark for comparing the performance of more advanced architectures.

DoubleDQN-PER-Dueling [8],[9],[10]: Combines Double Q-learning, Prioritised Experience Replay (PER), and Dueling Networks. This architecture

aims to improve value estimation stability, enhance sample efficiency during replay, and provide better separation between state and action value estimations. *DoubleDQN-PER-Dueling-Noisy [1]:* Extends the above architecture by incorporating Noisy Networks (NoisyNets). This modification introduces parametric noise into network weights to encourage directed exploration and adaptively balance exploration and exploitation. *DoubleDQN-PER-Dueling-NROWAN [6]:* Integrates Novel Random Weight Auxiliary Networks (NROWAN) into the DoubleDQN-PER-Dueling framework. This enhancement diversifies function approximation, improving stability and exploration in noisy environments. *Hierarchical Combinations:* Inspired by modularity research [1], [3], these architectures employ a hierarchical structure where high-level policies manage low-level DQN variants. These combinations aim to assess whether hierarchical control layers improve robustness and adaptability under challenging noise conditions.

C. Performance Evaluation

Performance is evaluated based on several key metrics, including:

1) Episodes to Solve

The number of episodes required to reach the task's defined success threshold. Evaluates how efficiently the architecture explores and exploit, since real world agents have limited chances to explore in real world applications.

2) Average Reward During Validation

After training, each model is tested for 500 episodes under a fixed moderate noise level. The average reward measures how well the agent retained knowledge of handling noise despite limited exposure, reflecting architectural robustness.

This methodology allows a comprehensive comparison of architectures, clarifying which components and modular frameworks yield the greatest resilience and adaptability to transient noise.

IV. Results

This section presents the key results obtained from the experiments on the CartPole-v1 and LunarLander-v3 environments. The evaluation focuses on two key metrics listed in Performance Evaluation. Figures 1–5 pertain to CartPole-v1, illustrating the average validation rewards, learning curves under different architectural configurations, and a robustness-efficiency analysis. Figures 6–10 mirror this analysis for LunarLander-v3. Figures 11 and 12 summarise the relationship between training episodes, validation rewards, and all tested architectures for each environment.

A. Episodes to Solve

1) CartPole-v1

In CartPole-v1, the number of training episodes required to reach the solution threshold varied notably across architectures. Standard DQN solved the environment in around 327 episodes. Certain hybrid approaches, such as DDPDNoisy + DQN, were even more efficient (228 episodes), highlighting how combining advanced exploration methods (NoisyNets) in high-level components with simpler low-level policies can expedite training under transient noise. Other combinations, like DDPDNoisy + DDPD and DDPDNROWAN + DQN, also demonstrated competitive training speeds. Overall, simpler architectures tended to achieve solution criteria quicker, while those incorporating more complex techniques (NROWAN, layered NoisyNets) sometimes required more episodes, reflecting a complexity-efficiency trade-off.

2) LunarLander-v3

For LunarLander-v3, the environment's greater complexity led to generally longer training times. Baseline DQN solved in approximately 956 episodes, whereas more intricate hierarchical combinations, especially those blending NROWAN and NoisyNets, could demand substantially more training time (e.g., DDPDNoisy + DDPDNROWAN took around 2692 episodes). While certain advanced architectures offered potential robustness benefits, they often incurred extended training durations. This suggests that the environment's complexity can amplify the overhead introduced by more complex architectural components.

B. Average Reward During Validation

1) CartPole-v1

Validation-phase average rewards (Figure 1) reveal how well architectures retain noise-handling strategies. While DQN achieved about 34.7 average validation reward, hybrid architectures often surpassed this baseline. DDPDNROWAN + DQN reached approximately 56.6, and DDPDNoisy + DDPD attained about 45.2. Such improvements highlight that incorporating NROWAN or NoisyNets, especially within a modular (hierarchical) framework, can enhance post-training robustness. However, not all complex architectures yielded uniformly higher validation rewards, underscoring the nuanced interplay among architectural components.

2) LunarLander-v3

In LunarLander-v3, the validation phase accentuated differences more sharply. While baseline DQN and DDPD variants hovered around 25–40 average validation rewards, architectures incorporating NoisyNets and NROWAN achieved significantly higher scores. DDPDNoisy alone reached 74.5, and DDPDNoisy + DDPDNROWAN achieved about

72.7 average validation reward. Although these enhanced architectures often traded off training speed, their ability to maintain robust performance during validation aligns with the study's objective of improving noise-handling capability.

V. Discussion

This section interprets the results considering our aim: understanding how modularity, NoisyNets, and NROWAN in hierarchical architectures influence robustness and efficiency. While some architectures excel in handling noise, they may require longer training or added complexity.

A. Impact of Noise on Performance

As anticipated, noise reduced stability and efficiency. While simpler architectures like DQN reached solution criteria relatively quickly in CartPole-v1, their post-training robustness was moderate. Conversely, more sophisticated architectures incorporating NoisyNets and NROWAN tended to better retain noise-handling knowledge, as evidenced by higher validation rewards in both environments. This suggests that complexity can foster more resilient internal representations of noisy conditions, albeit requiring more computational effort and training time.

B. Instability and Complexity Trade-offs

The introduction of advanced techniques improved robustness but did not guarantee smooth or rapid learning. Architectures with NoisyNets and NROWAN often demanded more episodes, especially in LunarLander-v3, to converge. Although these agents were ultimately more adept at performing under noise, the cost in training efficiency was evident. This trade-off highlights that there is no universal solution: practitioners must balance the need for robustness against available computational resources and acceptable training durations.

C. Analysis of Efficiency vs. Robustness

Figures 5 (CartPole-v1) and 10 (LunarLander-v3) present a quadrant-based analysis, plotting each architecture according to its learning efficiency (x-axis) and robustness (y-axis). The four quadrants represent: **Robust and Efficient (top-right):** Architectures that learn quickly and maintain high validation rewards. **Robust and Inefficient (top-left):** Architectures achieving strong noise-handling performance but requiring extended training times. **Less Robust and Efficient (bottom-right):** Architectures that solve quickly but fail to retain strong noise-handling strategies. **Less Robust and Inefficient (bottom-left):** Architectures that struggle both in speed and in post-training robustness.

In CartPole-v1, a few architectures managed to position themselves favourably in the robust and efficient quadrant, indicating that strategic combinations of hierarchical control and NoisyNets could yield balanced outcomes. However, many architectures ended up either trading robustness for efficiency or vice versa. Similar patterns emerged in LunarLander-v3, though fewer architectures reached the ideal quadrant due to the environment's complexity. This quadrant analysis integrates both performance metrics, reinforcing that architectural decisions must be tailored to the problem's complexity and training constraints.

D. Conclusion

In conclusion, this study demonstrates that while incorporating hierarchical modularity, NoisyNets, and NROWAN can heighten an agent's robustness to noise, these gains may come at the expense of extended training times or increased architectural complexity. By examining both efficiency and robustness, the analysis shows that no single architecture universally dominates. Instead, designers must carefully weigh their priorities rapid convergence, strong post-training stability, or resource constraints when selecting or engineering an RL architecture for noisy settings. Ultimately, the results highlight a nuanced design space. Agents can learn to handle noise effectively with the right architectural tools, but identifying architectures that achieve both robust and efficient learning remains an open challenge. Future work should focus on adaptive architectures, broader algorithmic variations, and more complex environmental conditions to better realise robust and efficient RL solutions in the face of real-world uncertainty.

E. Implications and Future Directions

The findings underscore that advanced architectural modifications can indeed enhance an agent's capacity to remember and leverage noise-handling strategies. Yet, complexity often entails trade-offs, such as longer training or partial diminishment of either robustness or efficiency. Future research might explore adaptive frameworks allowing agents to alter their architectural complexity as training proceeds or as noise patterns evolve. Such dynamic architectures could potentially avoid lengthy training plateaus or suboptimal allocations of computational effort. Experimenting with actor-critic methods and scaling to more realistic tasks and noise distributions would further clarify these dynamics. Additionally, incorporating sophisticated reward-shaping or curriculum learning approaches could guide agents towards mastering noise scenarios more swiftly and stably.

VI. References

[1] Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., ... & Blundell, C. (2017). Noisy networks for exploration. arXiv preprint arXiv:1706.10295.

[2] Hollenstein, J., Reymond, M., Calvano, M., & Pfister, J. P. (2022). Action noise in off-policy deep reinforcement learning: Impact on exploration and performance. arXiv preprint arXiv:2206.03787.

[3] Dulberg, Z., & Risi, S. (2022). Modularity benefits reinforcement learning agents with competing homeostatic drives. arXiv preprint arXiv:2204.06608.

[4] Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction. MIT Press.

[5] 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.

[6] Han, S., Liu, Y., Zhang, Y., & Li, G. (2020). NROWAN-DQN: A stable noisy network with noise reduction and online weight adjustment for exploration. arXiv preprint arXiv:2006.10980.

[7] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540.

[8] Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 30, No. 1).

[9] Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. arXiv preprint arXiv:1511.05952.

[10] Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In International Conference on Machine Learning (pp. 1995-2003).

VII. Appendix

This section includes the graphical results for the CartPole-v1 and LunarLander-v3 environments. The graphs are organised by performance metrics which are episodes to solve and average reward during validation and other insightful graphs relevant to the objective of the study.

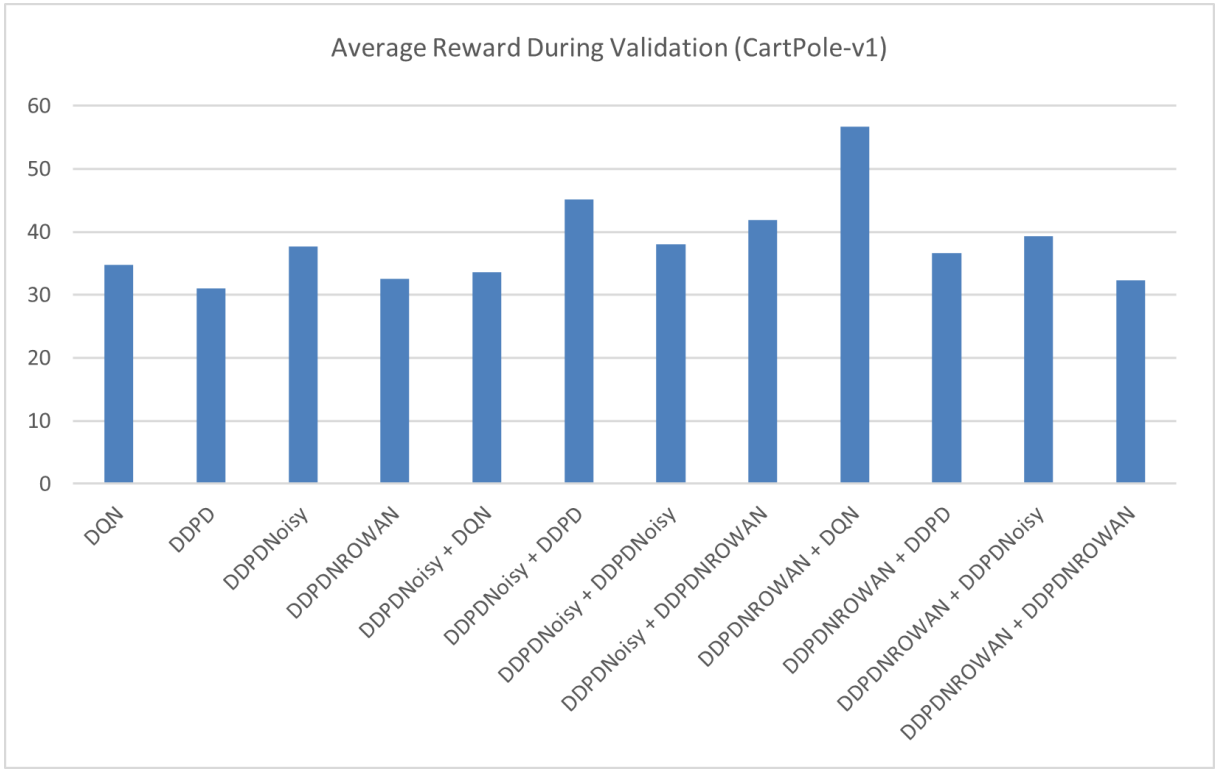

Figure 1: Average Reward During Validation(CartPole-v1)

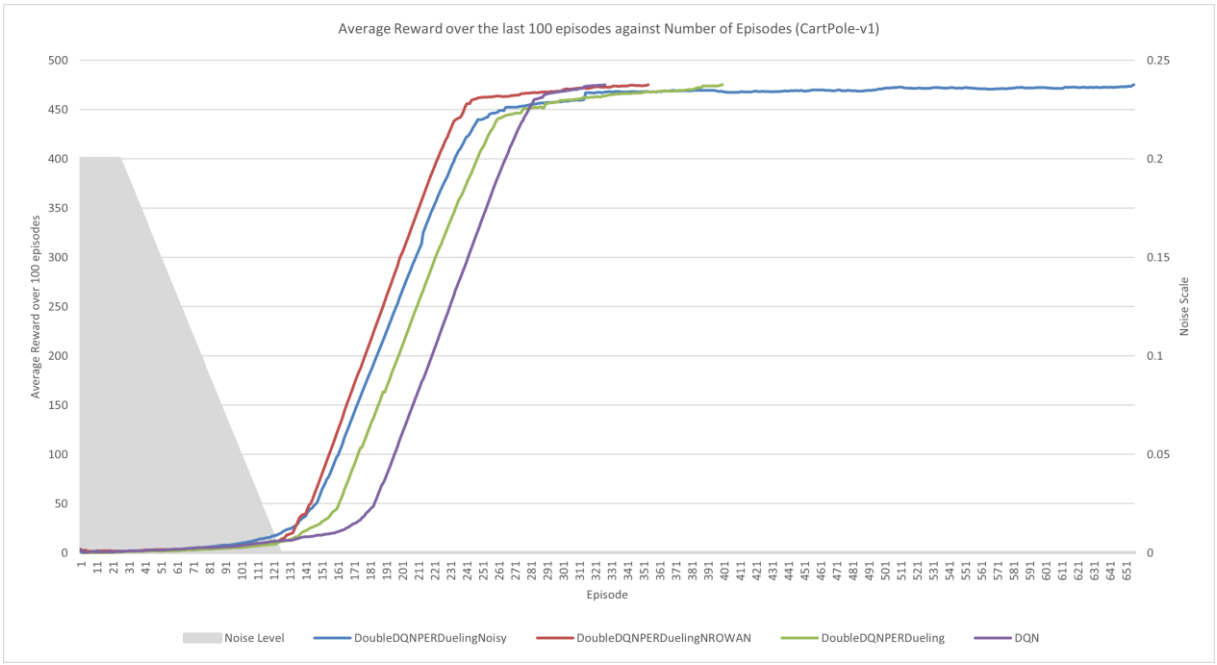

Figure 2: Average Reward over the last 100 episodes against Number of Episodes (CartPole-v1) for non-HRL architectures

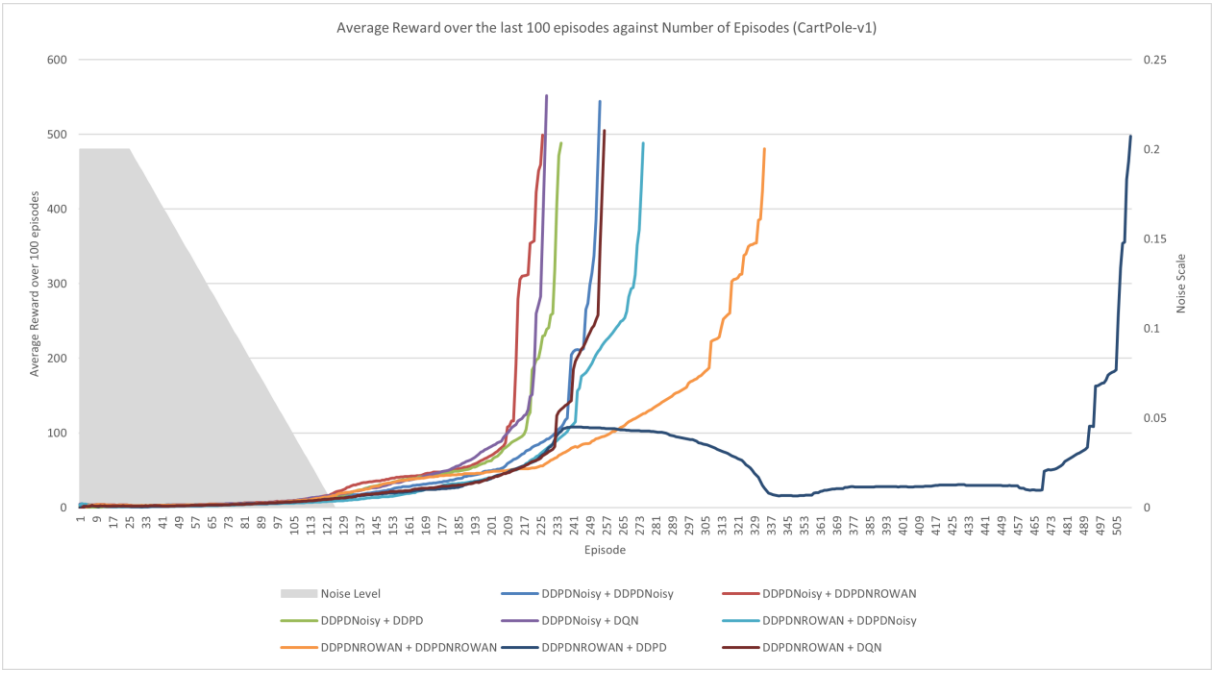

Figure 3: Average Reward over the last 100 episodes against Number of Episodes (CartPole-v1) for HRL architectures

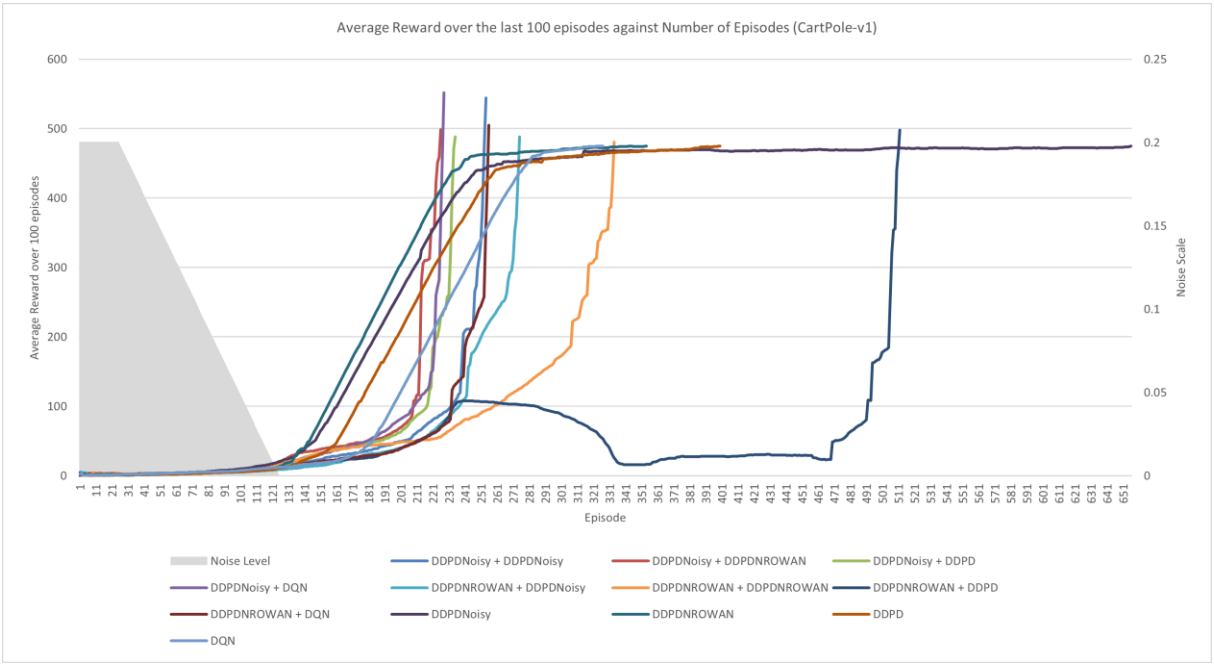

Figure 4: Average Reward over the last 100 episodes against Number of Episodes (CartPole-v1) for all architectures

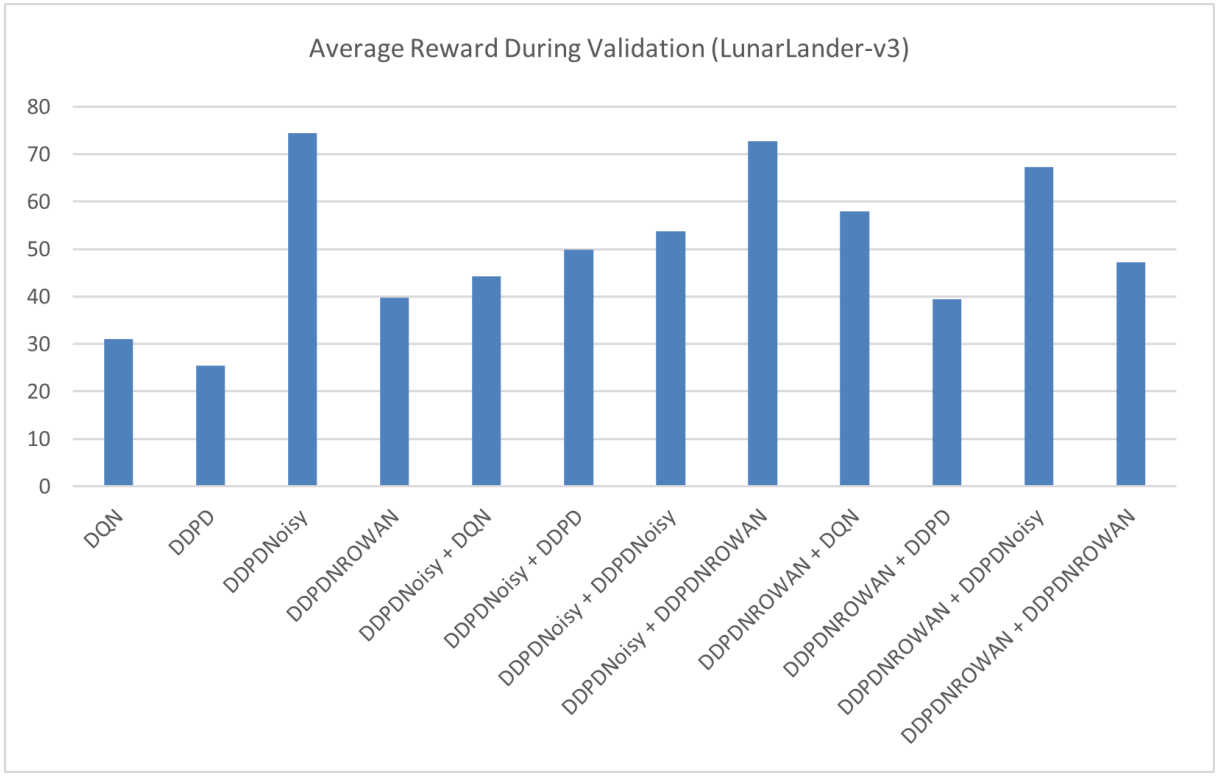

Figure 5: Average Reward During Validation (LunarLander-v3)

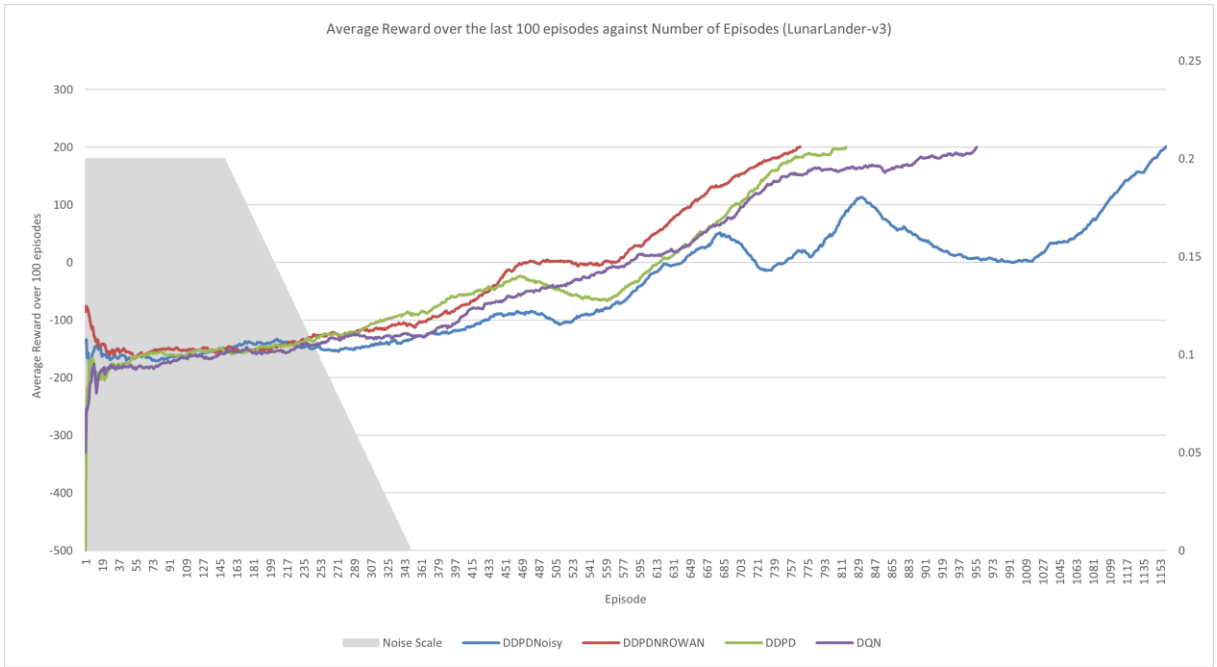

Figure 6: Average Reward over the last 100 episodes against Number of Episodes (LunarLander-v3) for non-HRL architectures

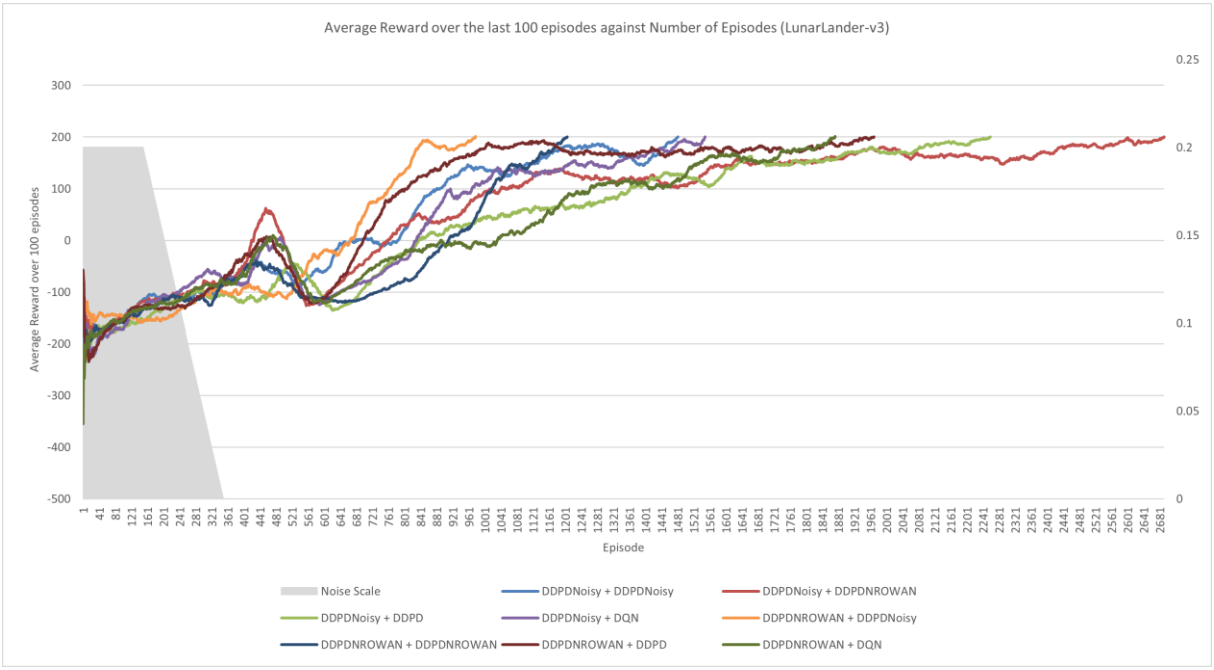

Figure 7: Average Reward over the last 100 episodes against Number of Episodes (LunarLander-v3) for HRL architectures

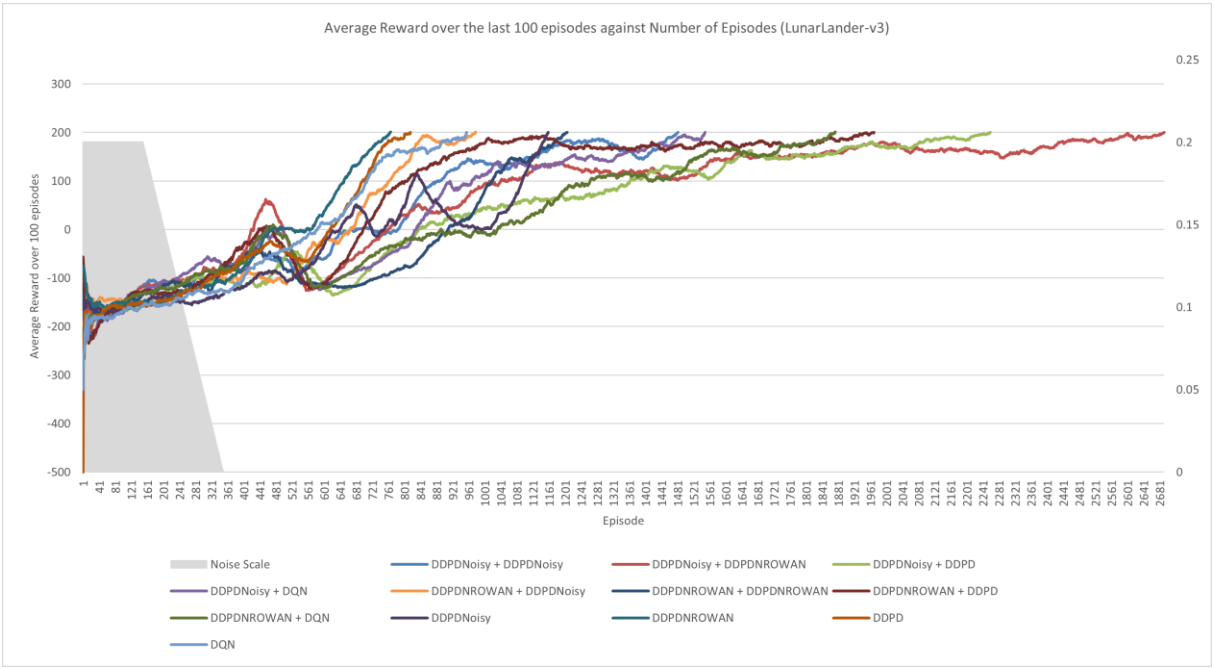

Figure 8: Average Reward over the last 100 episodes against Number of Episodes (LunarLander-v3) for all architectures

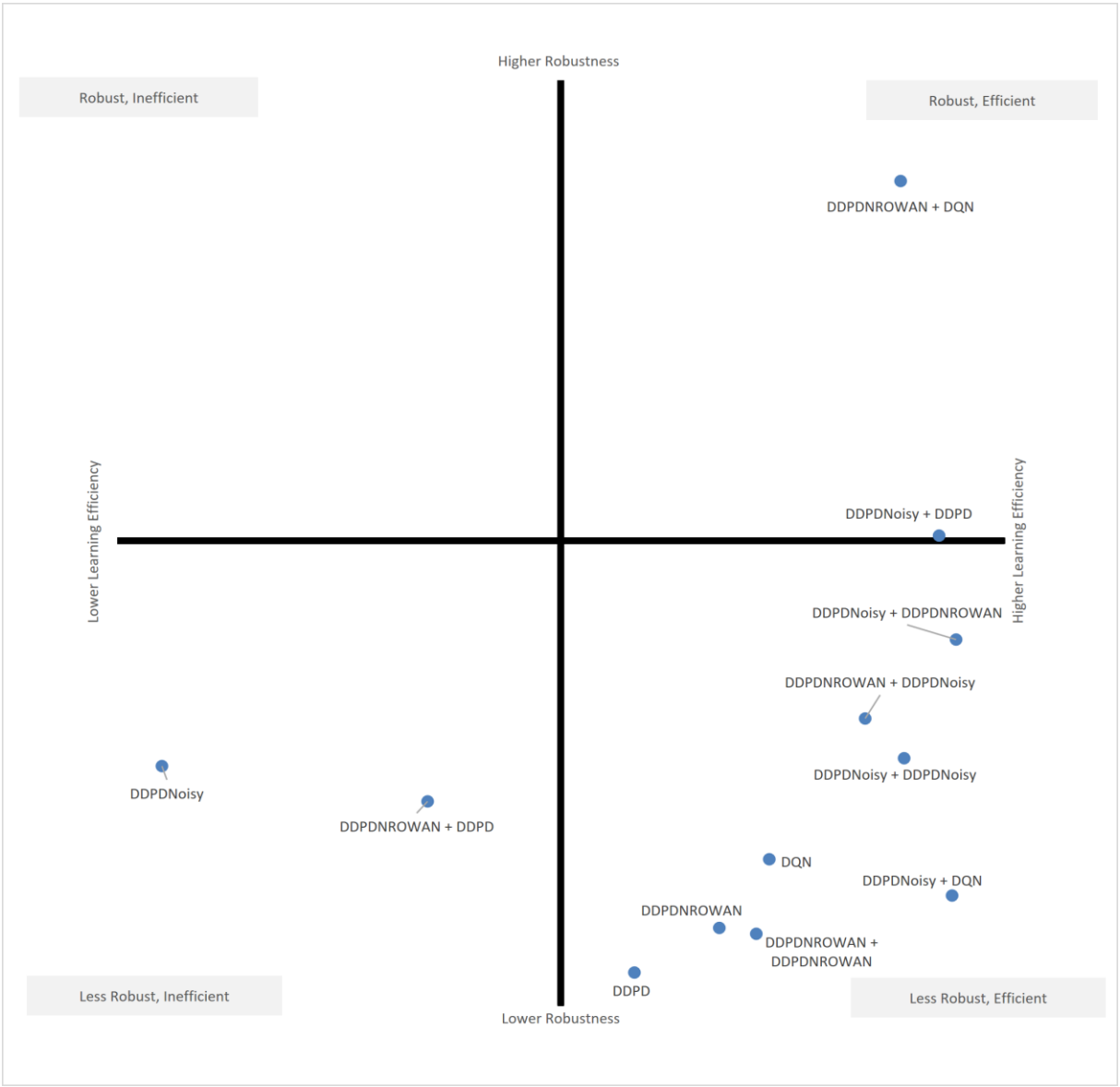

Figure 9: Robustness against Efficiency for all architectures(CartPole-v1)

Architecture	Episodes to Train	Average Reward During Validation
DQN	327	34.70530197
DDPD	400	31.04593213
DDPDNoisy	656	37.71316735
DDPDNROWAN	354	32.48636838
DDPDNoisy + DQN	228	33.52962631
DDPDNoisy + DDPD	235	45.165856
DDPDNoisy + DDPDNoisy	254	37.96914818
DDPDNoisy + DDPDNROWAN	226	41.81326841
DDPDNROWAN + DQN	256	56.63841393
DDPDNROWAN + DDPD	512	36.57630422
DDPDNROWAN + DDPDNoisy	275	39.25798016
DDPDNROWAN + DDPDNROWAN	334	32.29014562

Figure 10: Episodes to Train and Average Reward During Validation for all architectures(CartPole-v1)

Figure 11: Robustness against Efficiency for all architectures(LunarLander-v3)

Architecture	No of Episodes to T	Average Reward During Validation
DQN	956	31.0174859
DDPD	816	25.50261522
DDPDNoisy	1159	74.4614467
DDPDNROWAN	767	39.80889613
DDPDNoisy + DQN	1549	44.29681592
DDPDNoisy + DDPD	2260	49.87831866
DDPDNoisy + DDPDNoisy	1482	53.78687059
DDPDNoisy + DDPDNROWAN	2692	72.71211958
DDPDNROWAN + DQN	1873	58.02449997
DDPDNROWAN + DDPD	1970	39.46358739
DDPDNROWAN + DDPDNoisy	978	67.26338842
DDPDNROWAN + DDPDNROWAN	1206	47.15898157

Figure 12: Episodes to Train and Average Reward During Validation for all architectures(LunarLander-v3)