Exploring the Influence of Replay Buffer Size on Mitigating Performance Loss Caused by Environmental Variance

Ali Rafiei [1] **Reza Ghasemi** [1] **Uzair Sipra** [1] **Alexander Lambe Foster** [1]

Abstract

Deep reinforcement learning (RL) relies on Artificial Neural Networks (ANNs) to learn representations through function approximation. Deep Q-Network (DQN) is an RL algorithm that utilizes ANNs and experience replay buffers to approximate and optimize the Q-function, enabling efficient decision-making in sequential tasks. Despite their effectiveness, ANNs can be sensitive to high levels of variance from the returns they receive in an environment, which may impact performance. In this paper, we investigate whether adjusting the size of the replay buffer can mitigate performance fluctuations in n-step Deep Q-Networks (n-step DQN). Our results indicate that, under the right circumstances, increasing replay buffer helps mitigate the effects of environmental variance in returns on performance.

1. Introduction

Reinforcement Learning (RL) has garnered significant attention due to its versatility in solving a wide array of problems, ranging from autonomous systems and robotics to natural language processing (Li et al., 2016) and real-world applications, such as the work of Komorowski et al. (2018). RL enables agents to learn optimal decision-making policies through interaction with their environment, making it particularly suitable for scenarios involving complexity, uncertainty, and dynamic changes.

Artificial Neural Networks (ANN) have become prevalent in RL to obtain a close approximation for the value function of an optimal policy. A well known ANN-based algorithm that approximates action value function is Deep Q-Network (DQN) (Mnih et al., 2015; Mnih et al., 2013), which can be combined with n-step returns to form n-step DQN (Ostrovski et al., 2017).

[*]Equal contribution [1]University of Alberta.

CMPUT 655 Final Paper Submission

Problem statement: Although increasing the value of 'n' in n-step returns can provide a lower biased estimate of the value function, the trade off can be increased variance, potentially impacting performance. Recent research by Fedus et al. (2020) suggests that the replay buffer, which acts as a database storing historical transition data, might help mitigate this variance increase. The authors proposed that, given this property of replay buffers, it may be feasible to employ larger 'n' values, leading to improved performance. In light of this proposition, our investigation aims to explore the impact of replay buffers on the stability and performance of an agent when there exists variance in the environment. This raises our question: **Does replay buffer length mitigate performance impacts caused by variance in the returns of the environment.**

Understanding, mitigating, and regulating variance holds significant importance because uncontrolled variance can disrupt an agent's learning process, resulting in inadequate policy approximation. Numerous strategies have been proposed to manage variance, as evidenced by studies such as in Cheng et al. (2019); Anschel et al. (2017); Greensmith et al. (2001). These approaches encompass algorithmic analyses, neural network training stabilization techniques, regularization methods, and more. Our goal aligns with this pursuit, aiming to shed light on the relationship between replay buffer size and the performance impacts of environment's variance.

The rest of the paper has been organized as follows. The subsequent section will offer a brief overview of the background. Then the previous works will be presented in Section 3. In Section 4, we will delve into the practical implementation details. In Section 5 we will show the results of our experiments, and finally Section 6 we will discuss what the implications of our findings are, and what they suggest for future research.

2. Background

RL is a specific subset of machine learning dedicated to goal-oriented learning guided by rewards and trial-error search. An agent is placed in an unfamiliar environment and assigned the goal of maximizing the total reward it receives

through interactions with the environment, optimizing its decision-making process to identify actions leading to the highest reward. The agent must find a balance between exploiting its existing knowledge of the environment for immediate rewards and exploring new interactions with the hope of greater future rewards.

2.1. n-Step Q-Learning

An early breakthrough in RL was Q-Learning (Watkins, 1989), which is an off-policy temporal difference control algorithm. Q-learning makes one-step updates towards the optimal value function by directly computing the highest value function output over an observation using the maximizing action. For time t, state action value function Q, reward R_t, state s_t, action a, and discount factor γ, the Q-Learning update function is defined as:

$$Q_{t+1}(s_t, a_t) \leftarrow Q_t(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)].$$

Q-learning has been demonstrated to converge to the optimal policy with a probability of 1 at the limit, provided that all actions are repeatedly sampled in all states, and the action values are discretely represented when the Q function is in tabular form (Watkins & Dayan, 1992).

n-step Q-learning generalizes the standard Q-learning update function by truncating the full return after n-steps and then correcting for the remaining full return with $\max_a Q(s_{t+n}, a)$:

$$Q_{t+n}(s_t, a_t) \leftarrow Q_{t+n-1}(s_t, a_t) + \alpha\Big[G_{t:t+n} - Q_{t+n-1}(s_t, a_t)\Big],$$

where

$$G_{t:t+n} = \sum_{k=0}^{n-1} \gamma^k R_{t+k+1} + \gamma^n \max_a Q_{t+n-1}(s_{t+n}, a),$$

with intermediate actions generated according to the behavior policy.

2.2. Experience Replay

Experience replay (replay buffer) (Lin, 1992) is a critical component of DQN-style algorithms, serving as a memory bank of past agent transitions. The primary reason for using experience replay is to improve the stability and efficiency of the learning process. As the agent experiences the environment, the states, actions, and rewards are logged. The agent can then sample a randomized batch of transition data, eliminating the sequential correlation between transitions and ensuring the network can capture generalized feature semantics.

Empirically, experience replay allows large neural networks to train while avoiding many of the previous pitfalls of other stable training methods, such as needing to restart training repeatedly (Mnih et al., 2015). Batches and accumulated gradients from unbiased samples give the network a better approximation of the loss landscape, leading to better results.

2.3. Deep Q-Network

As the number of states in an environment increases, employing tabular RL methods becomes less feasible due to escalating computational requirements. Utilizing function approximation methods to model value functions can mitigate this limitation. One of the most powerful classes of function approximators are neural networks, and when used to approximate the action value function $Q(s, a)$, they give rise to many algorithms, one of them being the Deep Q-Network (DQN) architecture (Mnih et al., 2015; Mnih et al., 2013). The DQN training process involves two major components. Firstly, there is the experience replay buffer, the previously mentioned memory bank, where the agent stores and samples past interactions. Secondly, DQN utilizes two neural networks: the primary 'online network' which guides the agent's actions, and the 'target network,' which provides the target Q-values for learning. The target network is a delayed replica of the online network, with its parameters updated less frequently to maintain a degree of stability during training. A standard loss function for DQN is defined as follows:

$$Loss = \mathcal{L}(R_i + \gamma \max_{a'} Q(s_{i+1}, a'|\theta^-) - Q(s_i, a_i|\theta)), \quad (1)$$

Where $\mathcal{L}$ represents the smooth L1 loss implemented in PyTorch, as described in Girshick (2015); θ^- represents the target weights parameterizing the future state-action function; and θ represents the weights undergoing improvement through gradient descent, parameterizing the current state-action function. As DQN employs a replay buffer D, when computing the loss, it samples N entries uniformly: $(s_i, a_i, R_i, s_{i+1})^N \sim U(D)^N$. It uses the loss function to compute the error of each sample, then averages the error for back-propagation along the weights of the network that's learning. As stated previously, only the weights in θ are updated.
Figure 1 provides a high-level workflow diagram for DQN.

2.4. N-Step DQN

Standard DQN implements a single-step Q-Learning update. We augment DQN in the same manner as (Fedus et al., 2020), incorporating n-step updates from Rainbow (Hessel

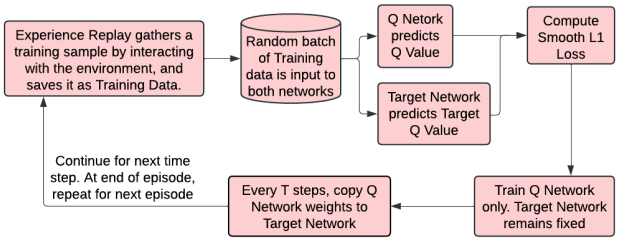

Figure 1. DQN Workflow Diagram (Doshi, 2020)

et al., 2018) onto the architecture. We thus keep track of additional reward observations in our replay buffer, and the loss function in [1] becomes:

$$Loss = \mathcal{L}(R_t^{(n)} + \gamma^n \max_{a'} Q(s_{i+n}, a'|\theta^-) - Q(s_i, a_i|\theta))$$

$$\text{where } R_t^{(n)} = \sum_{j=0}^{n-1} \gamma^j r_{t+j+1}.$$

3. Related Work

Previous works, such as that of Fedus et al. (2020), have explored the connection between experience replay size, n-step returns, and environmental variance. The authors hypothesized that the benefits from increasing the replay buffer capacity, with n-step returns, should be mitigated as the variance in returns decrease in an environment. However, we believe this hypothesis has been tested in a confounding manner. Those authors reported that the performance advantage of n-step returns in their ablation study diminished when sticky actions were removed. Sticky actions (Machado et al., 2018), which cause previously selected actions to have some chance of being selected again regardless of agent preference, increase variance of agent action selection which thus increases the variance of returns from the environment. Their hypothesis was tested by running a single experiment with different n-step lengths, and measuring the effect of different sizes of replay buffers on performance, leading to the conclusion "Turning off sticky actions reduces gains from increasing replay capacity" (Fedus et al., 2020).

While the disabling of sticky actions, which corresponds to a decrease in environmental variance, was shown to correlate with a decrease in performance with longer replay buffers, the authors neglected to consider that sticky actions have effects on agent behaviour that might extend beyond simply adding variance, such as the types of policies the agent can learn, among other issues. Furthermore, their experimental design lacked extensive testing of the hypothesis. Our work aims to fill this gap in conclusive results, by measuring the performance of an agent using n-step returns with different replay buffer sizes in multiple environments with various levels of environmental variance. To the best of our knowledge, no other work has examined our research question before, presenting an opportunity for us to further explore the validity of the claimed hypothesis via a series of experiments.

4. Experimental Design

Our experiments used the Deep Q-Network (DQN) (Mnih et al., 2015) architecture augmented with n-step returns, as demonstrated in the work of Fedus et al. (2020). We tested the agent's performance in the Gymnasium (Towers et al., 2023) RL test suite, specifically with the Frozen Lake and Lunar Lander tasks.

We devised a systematic approach for our experimental trials. We segmented our experiments into a tree-like structure, where the top level consisted of different environments. Each environment branched off into different levels of variance in the reward V. Each distribution further branched out into multiple nodes, where each node represented a specific buffer size used for testing. Each terminal node signifies a specific environment, reward distribution, and buffer size testing configuration.

4.1. Network Architecture

The neural network used in our agent was a simple feed forward network with two hidden layers, each with a dimension of 128 and using the ReLU activation function. The input and output to and from the network were determined by the environment; for Lunar Lander continuous observations were directly utilized as input, where as in Frozen Lake, discrete observations were first encoded as one-hot vectors. The network and agent were only configured to output discrete actions.

4.2. Hyper-parameter Configuration

Hyperparameter tuning is crucial for optimal model performance, generalization, and convergence during training. These parameters, set before training and not learned from data, are vital for preventing overfitting or underfitting, efficient use of computational resources, and improved model robustness. Given the vast hyperparameter space, an exhaustive tuning timeline was unrealistic, leading to focused tuning on a select number of parameters.

- Training Length: We experimented with a wide range of training lengths to ensure the agent had enough time to effectively learn while not wasting computational resources on diminishing returns once learning was effectively complete. For the Lunar Lander environment, we selected a training length of 200K frames. Meanwhile, for the Frozen Lake environment, we selected a training length of 10K frames.
- Epsilon: Agent training starts with an epsilon equal

to 1 and decays at a rate equal to the inverse replay buffer size until it tapers off to a minimum of 0.1. For Lunar Lander, the choice of epsilon decay was set to the inverse of the buffer size to ensure necessary exploration occurred for a good variety of experiences to be stored in the replay buffer. For Frozen Lake, epsilon decay was set to a constant value of $1/1500$ for all trials; this ensured the agent was able to sufficiently explore to reach the goal state.

- Replay Buffer Batch Size: Assessing the impact of different batch sizes sampled from the replay buffer was essential to optimizing the model's learning efficiency. Our comprehensive comparison encompassed varying batch sizes for both Lunar Lander and Frozen Lake, including 16, 32, 64, and 128, revealing distinctive performance patterns. Detailed analysis highlighted batch sizes of 32 and 64 as the optimal performers, exhibiting efficiency in information extraction while minimizing redundancy. Consequently, our preference for a batch size of 64 was founded on its ability to streamline learning by uniformly sampling experiences from the replay buffer.

- Optimizer: The choice of optimizer significantly influences a model's convergence time and learning efficiency. Guided by empirical comparisons (Choi et al., 2020), our focus narrowed down to RMSProp (Tijmen Tieleman, 2012) and ADAM (Kingma & Ba, 2014). Subsequent evaluation led us to select ADAM, configured with a learning rate of $1e-4$. This setup consistently demonstrated superior performance.

4.2.1. Environments

We chose to test our hypothesis on a simple environment such as Frozen Lake and on a more complex environment such as Lunar Lander, to see how the different levels of variance in the reward would affect the performance of an agent in a simple environment vs a complex environment. To test across a variety of popular RL benchmarks, we implemented an interface for the Gymnasium environment (Towers et al., 2023), a fork of the OpenAI Gym. To inject variance into the environment, at every time step, the return from the environment, R_e, was transformed into the agent's reward, R_a, by sampling from a normal distribution:

$$R_a \sim N(R_e, V \cdot R_e), \tag{2}$$

where V was fixed to some value between zero and two for a given experiment. The standard deviation for the distribution was capped at a minimum value of $1\mathrm{e}{-4}$ for $V > 0$. For both environments, agent performance across all combinations of $V \in [0.0, 0.6, 1.2, 2.0]$ and a selection of replay buffer lengths specific to each environment were evaluated. The performance of the agent was measured using cumulative reward over individual episodes; for every 500 (Lunar Lander) or 100 (Frozen Lake) training steps, the agent's learning was frozen, and the average cumulative reward was taken across three episodes.

Frozen Lake

In Frozen Lake, the agent is tasked with navigating a slippery Frozen Lake from a start position to a goal position without falling into any holes. Due to the slippery nature of the Frozen Lake, the agent may not always move as intended. Each observation received contains information on the agent's current position as:

$$\text{current row} \times \text{number of rows} + \text{current column} \tag{3}$$

Where both the row and column start at zero. For example, a 5x5 grid has 25 observations, and an arbitrary goal position on the bottom left-most tile would be calculated as: $4 * 5 + 0 = 20$. The agent must signal a left, right, up, or down action at every frame, receiving a reward of one if it reaches the goal and zero otherwise.

We chose this environment because of the unique challenge where the agent may not always move in its intended direction due to the complex environment and therefore receives a reward of zero in most cases. The environment allows us to gather meaningful performance data throughout the agent's learning process for replay buffer lengths $RB \in [200, 500, 1000, 1500, 2000]$. In addition, Frozen Lake allowed us to gather many trials due to the low computational demand and simplicity of the environment, unlike Lunar Lander.

Lunar Lander

The Lunar Lander environment was chosen as it is a slightly more complex task than Frozen Lake and, more importantly, one that has varying/more immediately informative levels of reward. The agent must pilot a small craft onto a landing pad by choosing to fire any of the craft's left, right, or centre orientation engines. Instead of receiving a reward of one for every time step, in the Lunar Lander task, the agent receives a variable positive or negative reward based on the distance from the landing target, the speed at which it moves, and whether the engines are firing or not, and more. With the Frozen Lake task, the reward is always centered on one or zero. With Lunar Lander, we get a better insight into how the agent is affected when the amount of reward is more influenced by its environment and immediate behaviors. We tested over replay buffer lengths, $RB \in [5k, 10k, 30k, 50k]$.

The action space, observation space, and rewards for Lunar Lander are as follows:

- Action Space (4 Discrete Actions): Do nothing, fire left orientation engine, fire main engine, fire right ori-

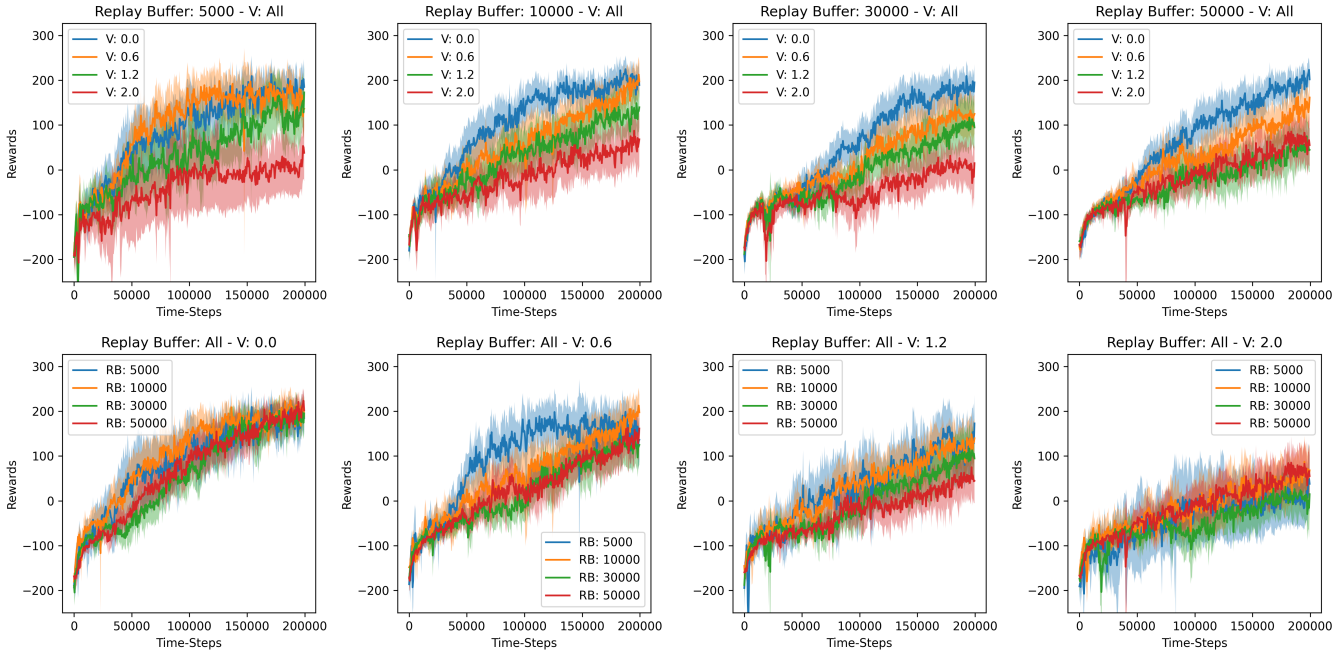

Figure 2. Results for Lunar Lander. **Upper**: All variance levels graphed together, per replay buffer length. **Lower**: All replay buffer lengths graphed together, per each variance level. The bold lines are averages over 18 independent learning trials. Every 500 time steps, the agent was tested for ten runs and the average reward was plotted. Shaded areas represent a 95% confidence interval. Figure 4 in the appendix shows the individual runs for each configuration.

entation engine (Klimov).

- Observation Space (8-dimensional vector): The coordinates of the lander in `x` & `y`, its linear velocities in `x` & `y`, its angle, its angular velocity, and two booleans that represent whether each leg is in contact with the ground or not (Klimov).
- Rewards: For moving from the top of the screen to the landing pad and coming to rest is about 100-140 points. If the lander moves away from the landing pad, it loses reward. If the lander crashes, it receives an additional -100 points. If it comes to rest, it receives an additional +100 points. Each leg with ground contact is +10 points. Firing the main engine is -0.3 points each frame. Firing the side engine is -0.03 points each frame. Solved is 200 points (Klimov).

5. Results

In this section we present the experimental results for the Lunar Lander and Frozen Lake environments.

5.1. Lunar Lander

Starting with Figure 2, the upper plots directly compare each replay buffer under different levels of variance. As one might expect, performance tends to drop as variance increases. Some artifacts appear, such as higher-variance experiments overtaking lower-variance experiments, but they are not statistically significant.

The lower set of plots in Figure 2 demonstrates that the performance across replay buffer sizes is surprisingly simi-

Performance (Average Reward) (Lunar Lander)				
Buffer Length	V = 0	V = 0.6	V = 1.2	V = 2
5k	181.6 ± 13.0	154.9 ± 14.6	137.5 ± 19.0	20.5 ± 19.1
10k	194.1 ± 10.0	194.3 ± 12.4	130.0 ± 15.0	61.8 ± 18.0
30k	183.4 ± 10.8	120.4 ± 14.5	103.7 ± 16.6	-0.4 ± 17.5
50k	202.0 ± 9.0	141.9 ± 13.6	52.2 ± 16.0	60.6 ± 16.2

Table 1. Final performance per configuration. Performance was measured using the average reward over the last 10 evaluations of each run, equivalent to the last 5K time steps of training

Change in Performance (%) (Lunar Lander)			
Buffer Length	V = 0.6	V = 1.2	V = 2
5k	-14.7 ± 15.2	-24.3 ± 17.6	-88.7 ± 17.7
10k	0.1 ± 11.5	-33.0 ± 12.9	-68.2 ± 14.4
30k	-34.3 ± 13.8	-43.5 ± 14.9	-100.2 ± 15.4
50k	-29.7 ± 11.2	-74.1 ± 12.4	-70.0 ± 12.5
Corr:	-0.7	-1.0	0.1

Table 2. Change in final performance per configuration relative to the baseline, $V = 0$. Includes the correlation between performance and replay buffer size for each entry. Performance was measured using the last 10 evaluations of each run, equivalent to the last 5K time steps of training

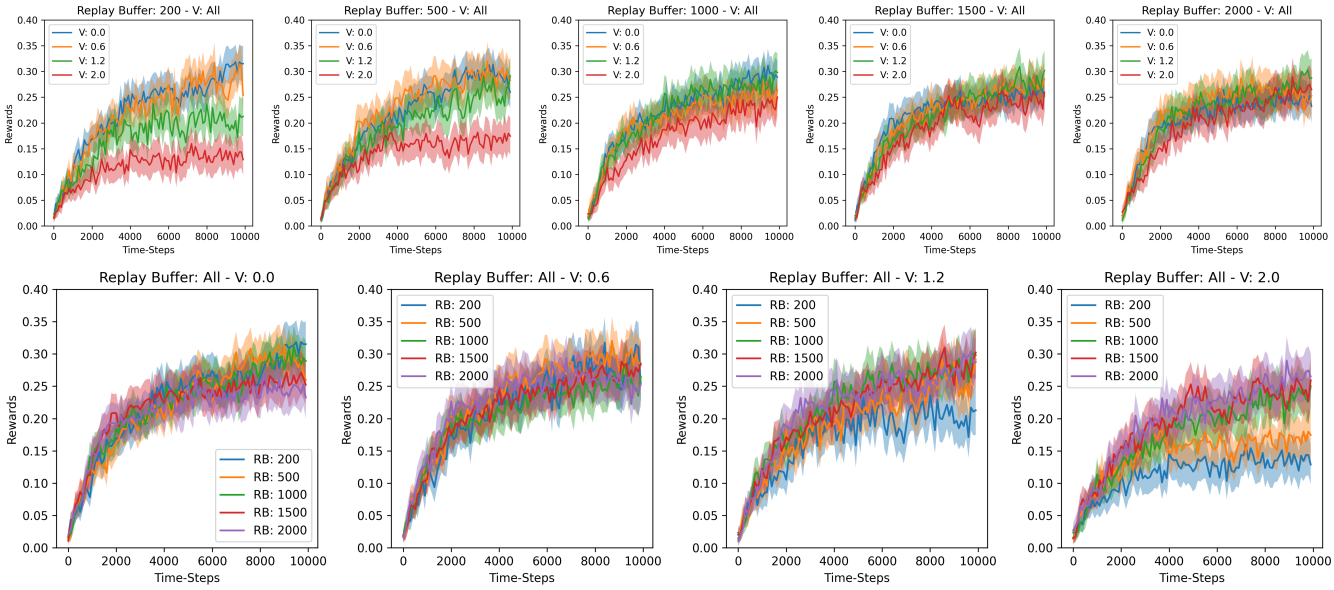

Figure 3. Results for Frozen Lake. **Upper**: All variance levels graphed together, per replay buffer length. **Lower**: All replay buffer lengths graphed together, per each variance level. The bold lines are averages over 238 independent learning trials. Every 100 time steps, the agent was tested for ten runs and the average reward was plotted. Shaded areas represent a 95% confidence interval. Figure 5 in the appendix shows the individual runs for each configuration.

lar. We can see that each curve clusters closely around the same range of final rewards and that each seems to be similarly affected by variance. Some interesting artifacts again appear in the middle of the learning curves, especially for the experiment with $RB = 5k$ and $V = 0.6$. The smaller replay buffer overtakes all the other entries before falling back in line towards the end. Additionally, all experiments for $RB = 10k$ out preform the equivalent experiments for $RB = 30K$ and $RB = 50K$, though it is not statistically significant across any specific trial.

To get a better idea of how well the agent succeeded in learning under each configuration, the average reward within a 95% confidence interval was computed across the final ten performance evaluations and plotted in Figure 1. For $V > 0$, $RB = 30k$ seems to perform overly poorly, failing to keep pace with the other experiments. Conversely, $RB = 10k$ performs quite well. For the highest level of variance, $RB \in [10k, 50k]$ appears to perform similarly well, while $RB \in [5k, 30k]$ perform similarly poorly.

Finally, Table 2 shows the percentage change in performance in Table 1 for each configuration, using $V = 0$ as a baseline. Additionally, the correlation between the buffer length and change in performance was calculated for each value of V.

Performance (Average Reward) (10x) (Frozen Lake)				
Buffer Length	V = 0	V = 0.6	V = 1.2	V = 2
200	3.13 ± 0.11	2.80 ± 0.12	1.97 ± 0.11	1.37 ± 0.10
500	2.87 ± 0.10	2.89 ± 0.12	2.58 ± 0.12	1.71 ± 0.11
1000	2.93 ± 0.10	2.56 ± 0.11	2.81 ± 0.12	2.36 ± 0.11
1500	2.61 ± 0.11	2.75 ± 0.11	2.79 ± 0.12	2.46 ± 0.11
2000	2.43 ± 0.10	2.50 ± 0.11	2.86 ± 0.12	2.65 ± 0.11

Table 3. Final performance per configuration. Performance was measured using the average reward over the last 10 evaluations of each run, equivalent to the last 1K time steps of training. All performance values scaled up by 10x.

Change in Performance (%) (Frozen Lake)			
Buffer Length	V = 0.6	V = 1.2	V = 2
200	-10.5 ± 7.2	-36.9 ± 7.0	-56.3 ± 6.7
500	0.5 ± 7.8	-10.1 ± 7.7	-40.4 ± 7.3
1000	-12.5 ± 7.2	-4.3 ± 7.5	-19.6 ± 7.3
1500	5.6 ± 8.3	6.8 ± 8.5	-5.8 ± 8.3
2000	3.0 ± 8.7	17.9 ± 9.0	9.1 ± 8.9
Corr:	0.6	0.9	1.0

Table 4. Change in final performance per configuration relative to the baseline, $V = 0$. Includes the correlation between performance and replay buffer size for each entry. Performance was measured using the last 10 evaluations of each run, equivalent to the last 1K time steps of training.

5.2. Frozen Lake

The data for Frozen Lake follows much more clear trends compared to Lunar Lander. Starting with the upper plots

in Figure 3, we see again that all replay buffer lengths are affected to some degree by variance. For $RB \in [200, 500]$ we see very apparent drops in performance relative to the other replay buffer lengths, strong enough to be statistically significant. The upper charts show that a large spread of performance exists under the lower replay buffer size, and the spread diminishes as the replay buffer length increases. Additionally, there are no specific over- or under-preforming curves. There's a relatively clear trend towards better performance with larger replay buffer lengths, and the trend becomes clearer with higher levels of variance.

These trends are laid out even more clearly in Tables 3 and 4. While $V = 0$ shows no clear trend, all entries with $V > 0$ show a clear hierarchy of both higher performance with larger replay buffer lengths and reduced performance impacts from increasing variance. The correlation also shows a strong trend: as the variance increases, so does the replay buffer length as an indicator of performance.

6. Conclusions

Based on the data collected and the analysis we've done, there's a strong connection between replay buffer length and the ability to mitigate variance in returns for the Frozen Lake environment, but no connection for the Lunar Lander environment. For Frozen Lake, performance is somewhat affected by variance across all replay buffer sizes, but the greater the length, the more resilient against variance the agent becomes. For Lunar Lander, all experiments demonstrated that agents with different replay buffer lengths were similarly affected by added variance in the value of returns, and the correlation between the replay buffer size and the change in performance degradation under higher levels of variance appears to show no consistent trend.

It seems that when the agent's performance is highly correlated to the reward it receives at any time step, as in Lunar Lander, the replay buffer has little effect on the agent's ability to learn with respect to environmental variance. In environments where the reward does not regularly provide useful information to the agent, longer replay buffers can help identify the true reward signal from the noise in the environment.

This indicates that we should be especially careful when considering the purpose and function of the replay buffer for a given task. When the reward signal is information-rich, the extra computation for a larger replay buffer could be a waste of time or, worse, could actually harm the agent's ability to learn; such as in the Lunar Lander experiments, where longer replay buffer lengths sometimes preform worse. Past a point, however, it becomes useful to employ larger replay buffers to help mitigate variance in the environment. It seems possible that there exists a continuum based on the amount of variance in the environment and the information the reward provides to the agent.

6.1. Limitations and Future Research

As indicated by the strange performance of agents at $RB = 30k$ in Lunar Lander, there's room for more data and investigation across a larger range of replay buffer lengths, especially with the limited number of samples we have to compute the correlation. Samples are also not independent within the same run, so the evaluations using the last 10 performance samples will be biased. Perhaps replay buffers do help mitigate variance in Lunar Lander, but $RB = 5k$ is enough to leverage whatever useful information the agent can extract from the environment. On the other hand, perhaps the ability to mitigate variance only becomes possible with replay buffer lengths over 50K.

Future research could include many more trials (orders of magnitude more) and tests across a larger range of replay buffer lengths for a more granular analysis. It would also be interesting to analyze the effects of variance on performance when the size of the replay buffer has more of an effect on learning. Each length tested produced agents capable of solving the environment at roughly the same speed. Perhaps a difference in relative performance occurs when the agent struggles to learn the environment relative to agents with a larger replay buffer. Finding and testing that range of values would be another useful experiment.

While our exploration primarily focused on ADAM due to the limited resources available, potential investigation into alternatives like NADAM (Dozat, 2016) remains an avenue for future studies, offering further insights into optimizer efficacy.

A limitation of our experiments with Lunar Lander was our lack of computational resources. With our local hardware, each trial run took approximately 25 minutes, which means over the total 18 trials averaged over 5 runs each, it took approximately 42 hours for us to collect our data. Due to this bottleneck in our testing pipeline, it would consume a significant amount of time to even double or triple our collective data for Lunar Lander. Future efforts could involve supplementing our computational resources with cloud services to expedite the process and allow for more extensive data collection.

Overall, there remains a gap in knowledge surrounding the effect of the replay buffer on environmental variance. Our work begins to elucidate the impactful effect of replay buffer size on the performance of different environments. Our work suggests replay buffers have more impact on mitigating variance on simple environments, or perhaps environments

that provide little information to the agent through their reward, like Frozen Lake. Future work could delve deeper into the studying these connections.

7. Contributions

The artifact for our paper can be found in the following reference (GitHub). Reza focused on the abstract and introduction, Uzair on the background, Ali on discussion and related work, and Alex on the experimental design and results. Overall, all members also contributed to each section of the project in revision. No single part of the final paper/project was not repeatedly and thoroughly worked on by every member of the team.

References

Anschel, O., Baram, N., and Shimkin, N. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 176–185. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/anschel17a.html.

Cheng, R., Verma, A., Orosz, G., Chaudhuri, S., Yue, Y., and Burdick, J. Control regularization for reduced variance reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1141–1150. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/cheng19a.html.

Choi, D., Shallue, C. J., Nado, Z., Lee, J., Maddison, C. J., and Dahl, G. E. On empirical comparisons of optimizers for deep learning, 2020.

Doshi, K. Reinforcement learning explained visually (part 5): Deep q networks, step-by-step. *Medium*, Dec 2020. URL https://towardsdatascience.com/reinforcement-learning-explained-visually-part-5-deep-q-networks-step-by-step-5a5317197f4b.

Dozat, T. Incorporating nesterov momentum into adam. *In ICLR Workshops*, 2016.

Fedus, W., Ramachandran, P., Agarwal, R., Bengio, Y., Larochelle, H., Rowland, M., and Dabney, W. Revisiting fundamentals of experience replay. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3061–3071. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/fedus20a.html.

Girshick, R. Fast r-cnn. *arXiv preprint arXiv:1504.08083*, 2015. URL https://arxiv.org/abs/1504.08083.

GitHub. Project Report Repository. https://github.com/uhsipra/Exploring-the-Influence-of-Replay-Buffer-Size.

Greensmith, E., Bartlett, P., and Baxter, J. Variance reduction techniques for gradient estimates in reinforcement learning. In Dietterich, T., Becker, S., and Ghahramani, Z. (eds.), *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/584b98aac2dddf59ee2cf19ca4ccb75e-Paper.pdf.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018. doi: 10.1609/aaai.v32i1.11796. URL https://ojs.aaai.org/index.php/AAAI/article/view/11796.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. *arXiv e-prints*, art. arXiv:1412.6980, December 2014. doi: 10.48550/arXiv.1412.6980. URL https://ui.adsabs.harvard.edu/abs/2014arXiv1412.6980K.

Klimov, O. Gym Documentation: Lunar Lander. https://www.gymlibrary.dev/environments/box2d/lunar_lander/. (Accessed on December 15, 2023).

Komorowski, M., Celi, L. A., Badawi, O., Gordon, A. C., and Faisal, A. A. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature medicine*, 24(11):1716–1720, 2018.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Mach. Learn.*, 8(3–4):293–321, may 1992. ISSN 0885-6125. doi: 10.1007/BF00992699. URL https://doi.org/10.1007/BF00992699.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., and Bowling, M. Revisiting the Arcade Learning Environment: Evaluation protocols and

open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, March 2018. doi: 10.1613/jair.5699. URL https://doi.org/10.1613/jair.5699.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1312.5602, December 2013. doi: 10.48550/arXiv.1312.5602. URL https://ui.adsabs.harvard.edu/abs/2013arXiv1312.5602M.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Tijmen Tieleman, G. H. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 2012.

Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., Cola, G. d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Shen, A. T. J., and Younis, O. G. Gymnasium, March 2023. URL https://zenodo.org/record/8127025.

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge United Kingdom, 1989.

Watkins, C. J. C. H. and Dayan, P. Q-learning. *Machine Learning*, 8(3):279–292, May 1992. ISSN 1573-0565. doi: 10.1007/BF00992698. URL https://doi.org/10.1007/BF00992698.

A. Appendix

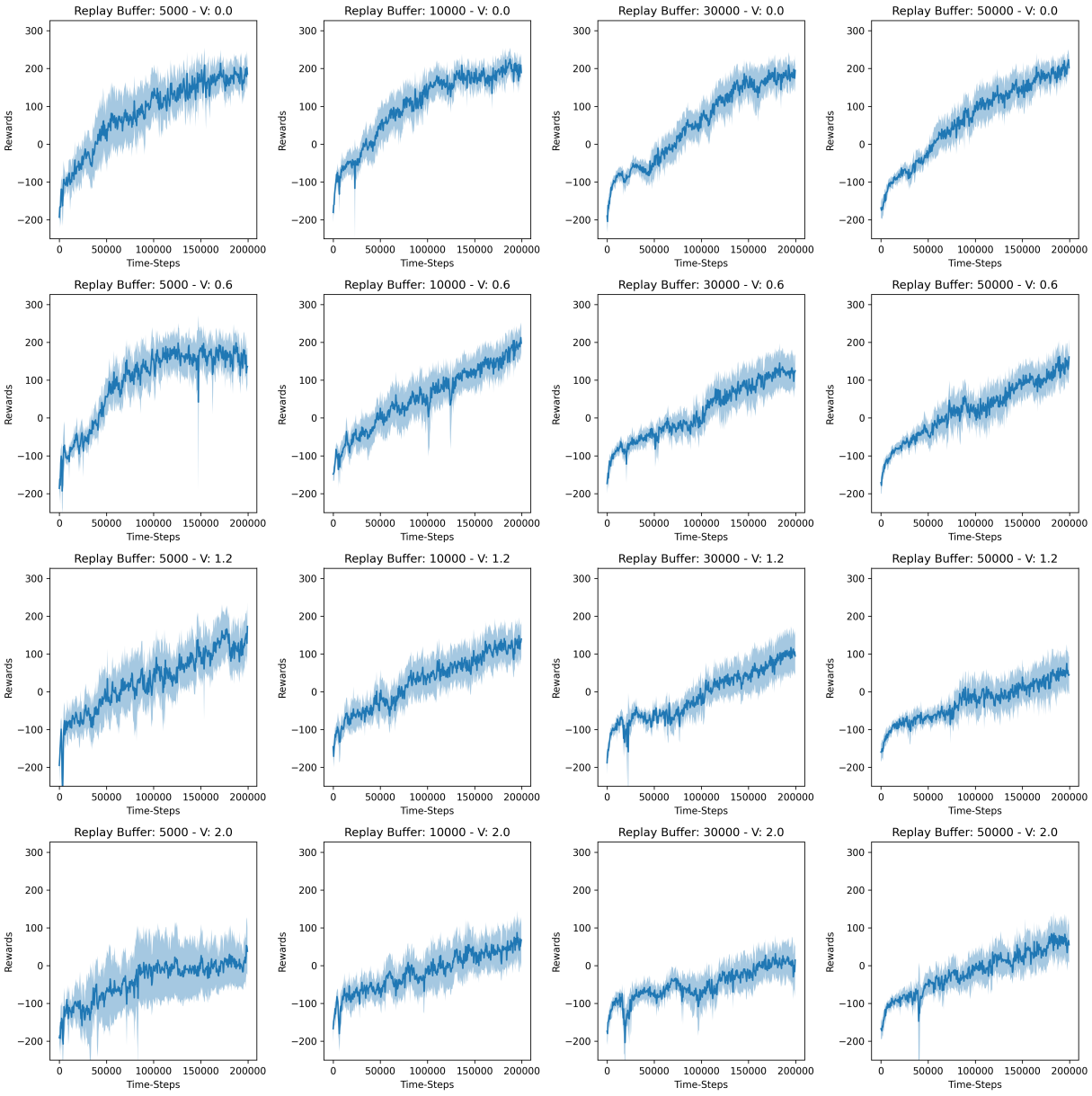

Figure 4. Average reward achieved by an agent in the Lunar Lander environment under varying sizes of the replay buffer and levels of variance The bold lines are averages over 18 independent learning trials. Every 500 frames, the agent was tested for ten runs and the average reward was plotted.

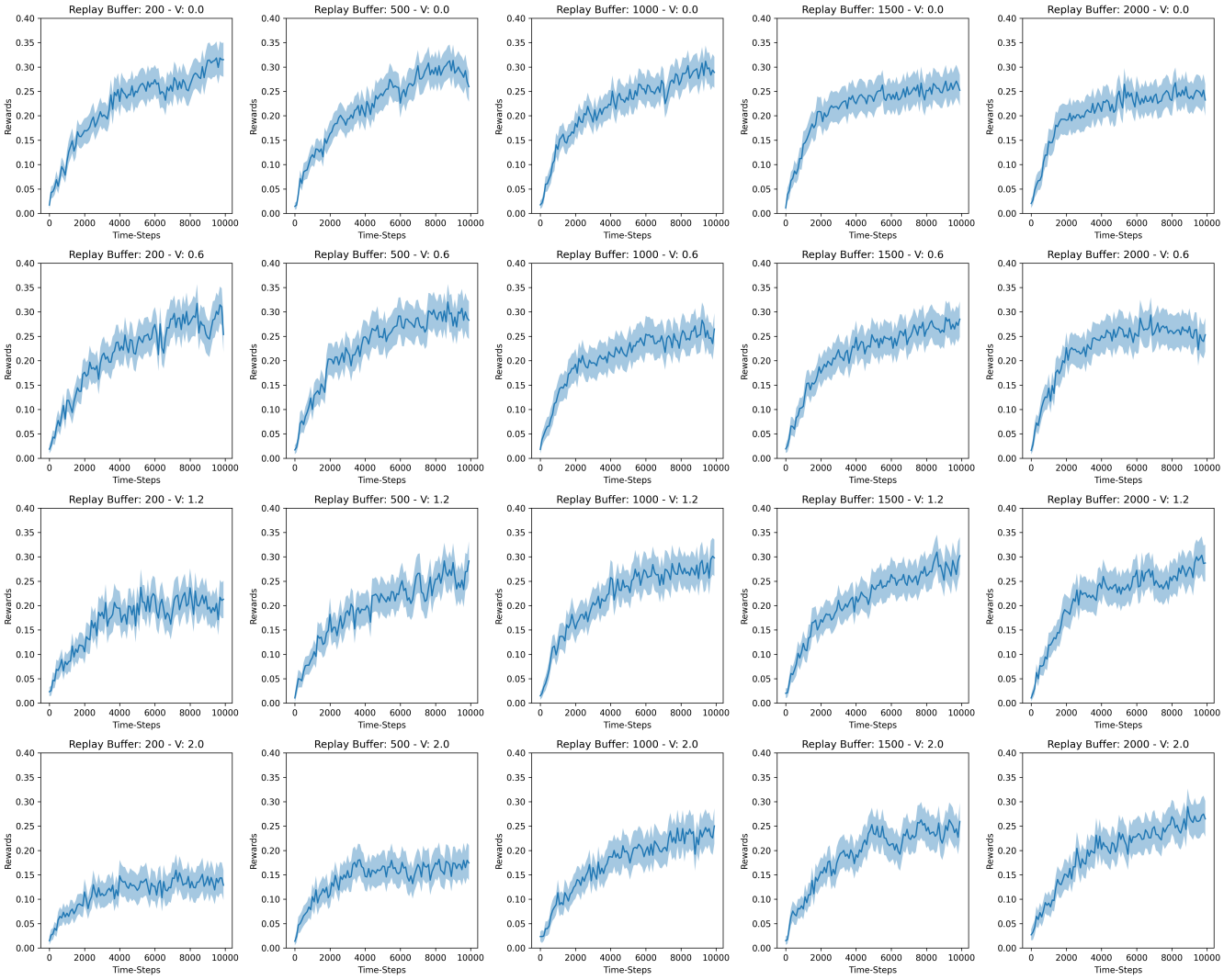

Figure 5. Average reward achieved by an agent in the Frozen Lake environment under varying sizes of the replay buffer and levels of variance The bold lines are averages over 238 independent learning trials. Every 100 time steps, the agent was tested for ten runs and the average reward was plotted.

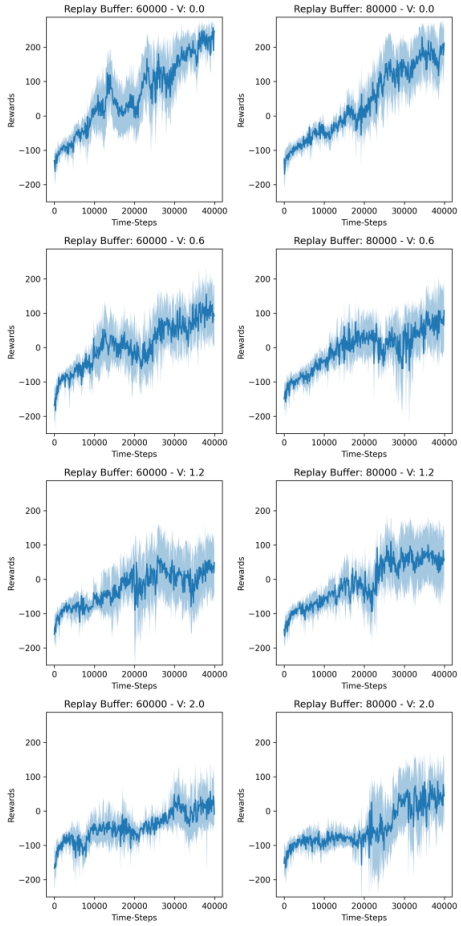

Figure 6. Average reward achieved by an agent in the Lunar Lander environment under 60k and 80k replay buffer and different levels of variance The bold lines are averages over 3 independent learning trials. Every 500 time steps, the agent was tested for ten runs and the average reward was plotted.

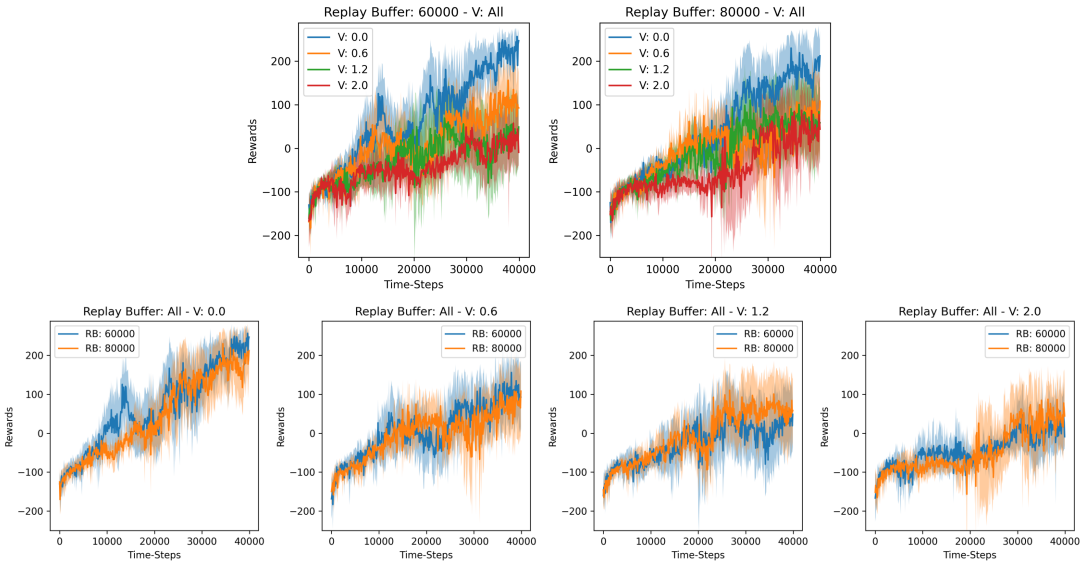

Figure 7. Results for Lunar Lander. **Upper**: All variance levels graphed together, per replay buffer length. **Lower**: All replay buffer lengths graphed together, per each variance level. Shaded areas represent a 95% confidence interval. This data was not used in our main results because we had only collected 3 trials worth of data for the 60K and 80K buffer configurations.