university of groningen / faculty of science and engineering / artificial intelligence

Comparing Sample Efficiency Between Model-Based and Model-Free Reinforcement Learning Methods

Bachelor's Project Thesis

Dennis Chong Yi Wu, s4752244, D.C.Y.Wu@student.rug.nl,
Supervisors: J.D. Cardenas Cartagena, M.Sc. & Dr M. Sabatelli

Abstract: Model-free Reinforcement Learning (RL) has been successfully applied to complex tasks, such as playing Atari games from image observations, but it typically requires a large amount of sample data. Model-based Reinforcement Learning attempts to address this issue by introducing a transition model, thereby reducing the number of samples needed to train an agent. This paper compares the sample efficiency of a Model-Free Double Deep Q-Network (DDQN) algorithm with a Model-Based Dyna-Q algorithm. Both agents were trained across various environments to determine which algorithm achieves a predefined reward threshold faster. Results indicate that while both agents can achieve the threshold, the model-based Dyna-Q algorithm consistently reaches it with fewer samples. However, this sample efficiency advantage comes at the cost of significantly higher computational resources.

1 Introduction

In the evolving field of machine learning, reinforcement learning (RL) stands out as a pragmatic methodology for enabling agents to make decisions and learn policies based on interactions with their environment. At its core, RL involves agents learning to achieve goals by taking actions that maximize the expected cumulative reward (discounted over time). In recent times the application of Deep Reinforcement Learning (DRL) on complex environments such as Atari games has become more prominent (Kaiser et al., 2020; Oh et al., 2015). This approach uses function approximation to predict expected rewards from taking an action. This process is heavily dependent on the agent's ability to sample data from its environment, where each sample consists of states, actions, and the resultant rewards (also known as a trajectory). However, the efficiency with which an RL agent can learn from these samples is a challenge, particularly in complex domains where obtaining each sample can be costly or time-consuming. Model-based Deep Reinforcement Learning (MBDRL) offers a possible solution to this by building up a transition model that mimics the dynamics of the environment by sampling from the environment. By using this transition model to generate artificial samples, the amount of sampling from the environment required can be reduced. Although model-based methods are a more sample-efficient way of doing reinforcement learning, the success of the model-based approach relies on the quality of the predictions of the dynamics model. Modelling the dynamics of high dimensional problems usually requires high-capacity networks that, unfortunately, require many samples for training to converge into the optimal policy, potentially outweighing the sample efficiency gains of model-based methods.
This paper aims to answer the following question:

Do model-based reinforcement learning agents need fewer samples compared to model-free reinforcement learning agents?

With the subquestions:

- How does the reward gain in model-based reinforcement learning compare to that of model-free reinforcement learning across varying buffer sizes?
- Are model-based agents able to gain higher rewards from the environment in comparison to model-free agents?

- Do model-based reinforcement learning agents require more training resources than model-free reinforcement learning agents?

The remainder of this paper is structured as follows: Section 2 will introduce important theoretical concepts of reinforcement learning, Section 3 will introduce the Methods used to answer the research questions, Section 4 will present the Results obtained from the experiments, and finally, Section 5 is a Conclusion and Discussion of the project.

2 Background

This section will provide a formal introduction to key concepts in reinforcement learning (RL). It will cover essential concepts in both model-based and model-free RL, including Markov decision processes, samples, and experience replay.

2.1 Markov Decision Processes

A Reinforcement Learning problem can be defined in terms of a Markov Decision Process with a state space $\mathcal{S}$, an action space $\mathcal{A}$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. The central problem in both Model-Free and Model-Based Reinforcement Learning is to find a policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ or value function V^π that maximizes the expected return $\mathbb{E}$ defined by Equation 2.1. Here $\mathcal{R}_t(s_t, a_t)$ is the reward received by the agent at time-step t after it takes an action a_t in state s_t. Furthermore γ^t is the Discount Factor which ensures that short-term reward gains are prioritized over those further in the future.. In model-free RL, this is done through direct interaction with the environment and learning from experience. In model-based RL, this involves building a model of the environment and using it for planning and decision-making. Understanding these distinctions and their implications helps in selecting the appropriate approach based on the specific characteristics and requirements of the problem at hand.

$$V^\pi(s_t, a_t) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}_t(s_t, a_t)\right] \tag{2.1}$$

2.2 Model-Free Reinforcement Learning

Model-Free Reinforcement Learning methods maximize the expected return using either policy-based or value-based methods. Policy-based approaches directly aim to learn the optimal policy that maximizes the expected return. Value-based approaches focus on estimating the value of states or state-action pairs to derive the optimal policy indirectly. Both policy and value-based techniques learn to make decisions without approximating the transition and reward functions of the MDP. They directly learn the optimal policy or value function through sample data from the environment. An example of a value-based method would be Q-learning (Watkins & Dayan, 1992). At the core, Q-learning is based around a $Q(s_t, a_t)$ function that provides Q-values that represent the expected return of taking a certain action a at a certain state s. It is updated towards an optimal Q-value function using the observed reward and the highest Q-value of the next state:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[\mathcal{R}_{t+1} + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \tag{2.2}$$

Here α is the learning rate and s_{t+1}, a_{t+1} are the state and taken action at the next time-step.

2.3 Model-Based Reinforcement Learning

In Model-Based Reinforcement Learning both $\mathcal{P}$ and $\mathcal{R}$ are approximated by the learned transition function $\hat{\mathcal{P}}(s_{t+1}|s_t, a_t)$ and the learned reward function $(\hat{\mathcal{R}}(s_t, a_t))$ respectively. After training $\hat{\mathcal{P}}$ and $\hat{\mathcal{R}}$ artificial samples are simulated through a process that is called planning as $(s_t, a_t, \hat{\mathcal{R}}(s_t, a_t), \hat{\mathcal{P}}(s_t, a_t))$. These artificial samples are then used to find the policy π or value function V^π by maximizing the expected return similar to model-free methods.

2.4 Deep Reinforcement Learning

As mentioned in Section 1 Deep Reinforcement Learning (DRL) agents have become more prominent recently for their capabilities to interact effectively with complex environments. Deep reinforcement learning aims to combine function approximators from machine learning with traditional re-

inforcement learning algorithms to make them scalable towards environments with large state spaces. By combining Q-learning with function approximators and a Experience Replay Buffer a Deep Q-network (DQN) (Watkins & Dayan, 1992) was created. It has proven to be able to perform well in a wide variety of game-based environments.

2.5 Experience Replay Buffer

When an agent interacts with an environment the state of the environment changes, this transition can be expressed as a set of elements (s_t, a_t, r_t, s_{t+1}) which is called a *sample*. Samples are also referred to as Experience or Trajectories are essential in efficiently training agents.

The Experience Replay Buffer (Lin, 1992) is a fixed-size storage container that holds samples collected from the environment. It is vital in the training process of both deep model-free and model-based agents since it provides the ability to reuse samples collected previously which leads to more stable training. The most common implementation of the Experience Replay Buffer uses a circular design. When the buffer is full, it removes samples on a first-in, first-out basis, meaning old samples are gradually replaced by new ones. Experience Replay Buffers also generally provide a way to sample batches of samples in a uniform way, which means that each sample in the buffer has an equal probability to be sampled.

3 Methods

This section will introduce the methods that are essential to this project. The section starts out by introducing the Double Deep Q-network and Dyna-Q algorithm, hereafter the experimental design will be introduced. Furthermore, the neural network architectures that were used for the experiments are outlined. Finally, the main differences across environments are shown.

3.1 DDQN

The agent chosen for this project is a Double Deep Q-Network (DDQN) (Van Hasselt et al., 2015) agent, as described by algorithm 3.1. It is an extension of the Deep Q-Network (DQN) framework introduced earlier. As its name suggests, a DDQN consists of two DQN networks: a primary network and a target network. The primary network is used to select actions, while the target network is used to provide stable Q-value targets for the training updates.

This separation helps mitigate the problem of overestimation of Q-values, which can occur in standard DQN implementations. By decoupling the action selection and Q-value target calculation, DDQN improves learning stability and leads to better overall performance. These advantages make it an ideal choice for the project.

Before training starts the replay buffer $\mathcal{D}$ is initialized to store the agent's experiences, and both the primary network Q_θ and the target network $Q_{\theta'}$ are initialized with weights θ and $\theta' = \theta$, respectively. In each episode, starting from an initial state s_0, the agent selects actions a_t using an ϵ-greedy policy derived from Q_θ. Actions are either random (with probability ϵ) or the ones that maximize the Q-value (with probability $1-\epsilon$). After executing action a_t, the agent observes the reward r_t and next state s_{t+1}, storing the transition (s_t, a_t, r_t, s_{t+1}) in the replay buffer $\mathcal{D}$. A random mini-batch of transitions (s_j, a_j, r_j, s_{j+1}) is sampled from $\mathcal{D}$. For each transition, the target value $Q_{\theta'}(s_j, a_j)$ is computed as:

$$a' = \arg\max_a Q_\theta(s_{j+1}, a) \tag{3.1}$$

$$Q_{\theta'}(s_j, a_j) \approx r_j + \gamma Q_{\theta'}(s_{j+1}, a') \tag{3.2}$$

A gradient descent step is performed to minimize the squared error between $Q_{\theta'}(s_j, a_j)$ and $Q_\theta(s_j, a_j)$. Every C steps, the target network weights are updated to match the primary network weights, $\theta' \leftarrow \theta$. This process repeats until the training ends or resources are exhausted.

3.2 Dyna-Q

The model-based algorithm that was used for the experiments in this project is a variation of the Dyna-Q algorithm first introduced by (Sutton, 1991) The key difference is that the Dyna-Q variant used for this project does not sample the state-action pair used for planning from a set of previously seen state-action pairs but instead is sampled from the Experience Replay Buffer. This is to ensure that the transition model simulates the

Algorithm 3.1 Double Deep Q-Network (DDQN)

Initialize replay buffer $\mathcal{D}$
Initialize primary network Q_θ with random weights θ
Initialize target network $Q_{\theta'}$ with weights $\theta' = \theta$
Set learning rate α, discount factor γ, batch size B, and target update frequency C
for episode = 1, M **do**
  Initialize state s_0
  for t = 1, T **do**
    Select action a_t using ϵ-greedy policy from $Q_\theta(s_t, a_t)$
    Execute action a_t
    Observe reward $r_t = \mathcal{R}(s_t, a_t)$
    Observe next state $s_{t+1} = \mathcal{P}(s_t, a_t)$
    Store sample (s_t, a_t, r_t, s_{t+1}) in replay buffer D
    Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
    Calculate target $Q_{\theta'}(s_j, a_j)$ for each transition in minibatch:

$$Q_{\theta'}(s_j, a_j) \approx \begin{cases} r_j \text{ if } s_{j+1} \text{ is terminal} \\ \text{otherwise:} \\ r_j + \gamma Q_{\theta'}(s_{j+1}, \arg\max_a Q_\theta(s_{j+1}, a)) \end{cases}$$

    Perform a gradient descent step on loss:

$$L(\theta) = \frac{1}{B} \sum_j \left(Q_{\theta'}(s_j, a_j) - Q_\theta(s_j, a_j)\right)^2$$

    Update weights θ using gradient descent:

$$\theta \leftarrow \theta - \alpha \nabla_\theta L(\theta)$$

    If $t \mod C == 0$:
    Update target network $Q_{\theta'} \leftarrow Q_\theta$
  end for
end for

Algorithm 3.2 Dyna-Q

Initialize $Q(s, a)$ arbitrarily
Initialize *model* as $\hat{P}(s_{t+1}|s_t, a_t)$, and $\hat{R}(r_t|s_t, a_t)$
Set simulation size n
for t = 1, T **do**
  Select and execute action a_t using ϵ-greedy policy from $Q(s_t, a_t)$
  Observe reward $r_t = \mathcal{R}(s_t, a_t)$
  And next state $s_{t+1} = \mathcal{P}(s_t, a_t)$
  Store sample (s_t, a_t, r_t, s_{t+1}) in replay buffer D
  Sample random minibatch of transitions (s_j, a_j, r_j, s_{j+1}) from D
  Update $Q(s_j, a_j)$
  Update $model(s_{t+1}, r_t|s_t, a_t)$
  for $i = 1$ to n **do**
    Sample (s_j, a_j) from buffer $\mathcal{D}$
    $r_j, s_j + 1 \leftarrow model(s_j, a_j)$ {planning}
    Update $Q(s_j, a_j)$
  end for
end for

environment instead of merely reproducing past samples. The algorithm begins by initializing the weights of the DDQN agent and training the transition model using data collected by an agent pretrained on the environment. In each iteration, the agent interacts with the environment, and these interactions are stored as samples in the Experience Replay Buffer. Subsequently, both the agent and the transition model were trained on a batch of 128 samples from the buffer. The algorithm then enters the planning stage, where the transition model simulates a batch of X samples that is used to further train the agent. This process repeats until training is completed.

3.3 Experimental Setup

To evaluate the sample efficiency of model-based and model-free approaches, experiments were conducted across the Cartpole and Acrobot environments, the Lunar Lander environment which is part of the Box2D benchmark and the Atari:Freeway environment which is part of the Atari Learning Environment (ALE)(Bellemare et al., 2012). All environments were accessed through OpenAI Gym (Brockman et al., 2016). Changes in hyperparameters and neural network architectures were made across environments, Section 3.5 and, Appendix A and Bwill go into more detail on this. Because the

experimental results were consistent across all environments, this paper will focus on a detailed analysis of the Cartpole environment for the sake of simplicity. All of the other experimental results can be found in the Appendix. Before starting the experiment, samples were collected by a trained model-free agent and stored in an Experience Replay Buffer. During the experiments both the model-based and the model-free agents interacted with the environment X amount of times. The experiment was repeated for 3 runs for every one of the 9 buffer sizes varying from 1000 to 9000 with an increment of 1000 every time. Every 50 interactions with the environment the network weights were updated. During the experiment, the total amount of (non-artificial) samples used, the performance of the agents over 3 episodes every 500 updates and the amount of weight updates were kept track of.

3.4 Network Architectures

All of the experiments were performed on both a Double Deep Q-network and an altered version of the Dyna-Q algorithm using a Double Deep Q-network (DDQN) as the agent as described by Section 3.2. The network parameters of the DDQN network as well as the parameters of the transition model are given by Appendix A.

3.5 Differences Across Environments

As mentioned before in Section 3.3 all of the experiments were performed on Cartpole, Acrobot, Lunar Lander and Atari:Freeway notable changes in hyperparameters between is an increase in training time as state complexity increases since it takes longer for the agent to learn the optimal policy. As well as a decrease in simulated data due to a limit in computational resources. All of the hyperparameters can be at Appendix A. Other notable differences between states is the DDQN agent's network as a regular Multi layer perception is swapped out for a Convolutional Neural Network when working with the Freeway environment as it has high dimensional pixel-based states. All DDQN network specifications can be found at Appendix B. The transition model used to simulate the Freeway environment is also different from the transition model used on the other environments as it needs to work with pixel images. The exact transition model specifications can be found at Appendix B.

4 Results

This section will comprehensively address all research questions by presenting the results obtained from the conducted experiments, which will subsequently be analyzed.

4.1 Sample Efficiency

To address whether model-based reinforcement learning (RL) agents require fewer samples compared to model-free RL agents, the mean number of samples needed to reach a certain threshold reward across different buffer sizes was kept track of. The threshold reward was different for each environment and was chosen to represent a level of performance deemed reasonable by the experimenter.

The results shown in Figure 4.1 reveal that model-based RL agents consistently needed fewer samples to achieve the threshold reward compared to model-free RL agents. This trend was observed across all tested buffer sizes. Model-based RL agents not only reached the threshold reward with fewer samples but also exhibited less variability in the number of samples required, as indicated by the smaller standard deviations. This implies a more consistent performance compared to model-free RL agents.

4.2 Buffer Sizes

To compare the reward gain of model-based reinforcement learning (RL) agents with that of model-free RL agents across varying buffer sizes, The mean reward over three episodes was tracked at intervals of every 500 update steps.

One of the key observations from comparing the reward gain of model-based reinforcement learning (RL) agents to that of model-free RL agents is the notable "jump-start effect" in the performance of model-based RL agents at the start of training. This phenomenon is characterized by an initial spike in performance for model-based RL agents.

The cause of this jump-start effect lies in the pre-training process. Before the main training phase begins, the transition model used by the model-based

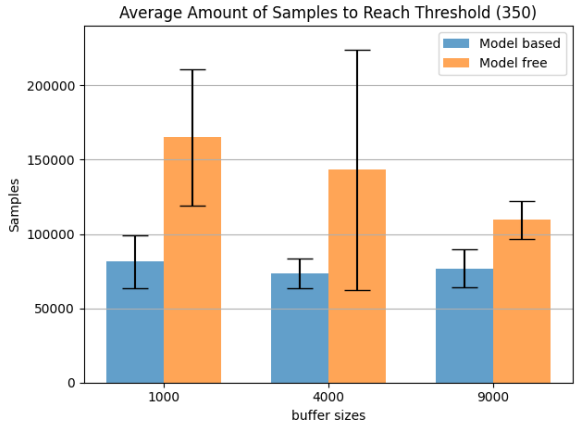


Figure 4.1: A comparison of the amount of samples needed for the model-based and model-free agents to reach a certain reward.

RL agent is pre-trained on existing data. During the main training phase, the model-based RL agent utilizes both data gathered directly from the environment and data generated by this pre-trained transition model. Since the agents are trained in batches, this jump-start effect becomes even more pronounced. Figures 4.2A, 4.2B and 4.2C all illustrate the initial performance boost experienced by the model-based RL agent compared to the model-free RL agent.

The results also indicate that increasing buffer sizes contribute to training stability for both model-based and model-free RL agents. Larger buffer sizes provide a more comprehensive dataset for training, which helps mitigate the variability in performance during the training process. This stabilizing effect is observed equally in both types of agents. Despite the increased stability with larger buffer sizes, the final performance of both model-based and model-free RL agents after training does not show significant improvement.

4.3 Post-training Performance

To determine if model-based reinforcement learning (RL) agents are able to gain higher rewards from the environment in comparison to model-free RL agents, the mean of the last rewards across different buffer sizes for both agents was tracked.

The results shown in Figure 4.3. indicate that there is no significant performance difference be-

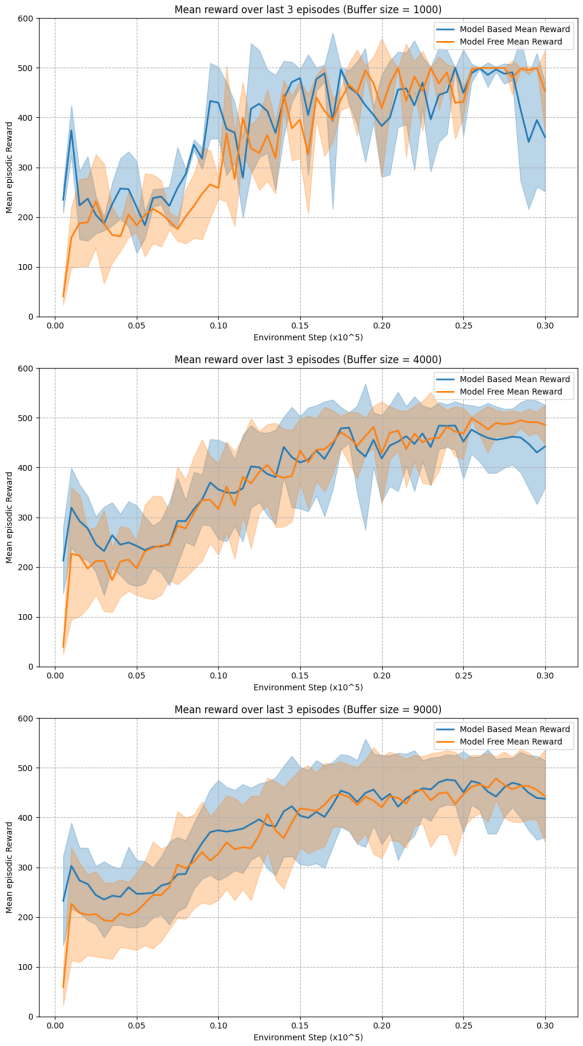


Figure 4.2: Comparison of the Model-based and Model-free agent's mean reward over 3 episodes every 500 update steps across buffer sizes.

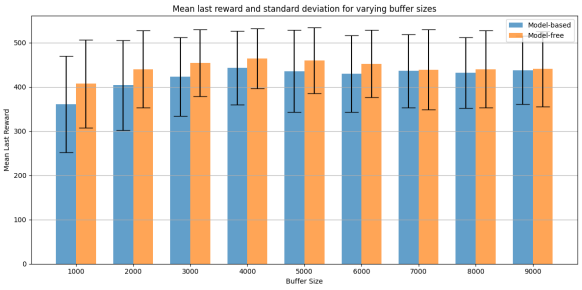


Figure 4.3: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

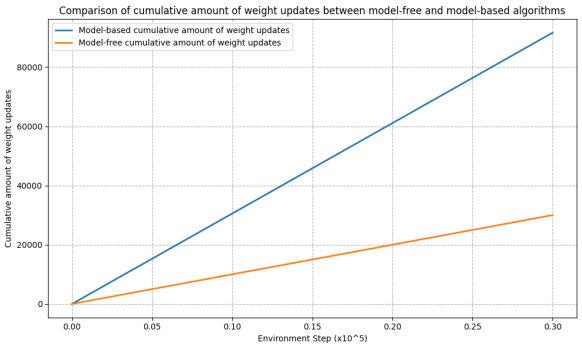


Figure 4.4: Comparison between the Model-based and Model-free agent's amount of weight updates.

tween model-based and model-free agents at the end of training. Both types of agents achieved similar rewards with very similar standard deviations across different buffer sizes. This suggests that, given sufficient training time, both model-based and model-free agents converge to similar policies

4.4 Resource Cost

To address whether model-based reinforcement learning (RL) agents require more training resources than model-free RL agents, the number of weight updates was tracked throughout the training process. Additionally, the total number of weights in the networks of both types of agents was compared.

The analysis reveals that model-based RL agents require significantly more training resources compared to model-free RL agents. Specifically, the model-based RL agents needed roughly three times as many weight updates as seen in Figure. This increased computational demand is illustrated in Figure 4.4. Moreover, the model-based RL agents had approximately three times as many weights in their networks compared to the model-free RL agents. This difference in the complexity of the networks contributes to the higher number of weights updated in the training of a model-based RL agent.

5 Discussion and Conclusion

This paper systematically compared the performance and resource requirements of model-based and model-free reinforcement learning (RL) agents across several criteria: sample efficiency, reward gain, final performance, and training resources.

- Sample Efficiency: The findings indicate that model-based RL agents required significantly fewer samples to achieve a predefined performance threshold compared to their model-free counterparts. This advantage was underscored by smaller standard deviations in the number of samples needed, suggesting more consistent performance across different environments. This efficiency is particularly advantageous in scenarios where data collection is costly or time-consuming.

- Reward Gain: The initial phase of training revealed a distinct jump-start effect for model-based RL agents. This phenomenon, attributed to the pre-training of the transition model on existing data, facilitated a rapid initial performance boost. However, as training progressed, both model-based and model-free agents demonstrated improved stability with larger buffer sizes.

- Final Performance: Despite the initial advantage, final performance levels in terms of mean rewards converged between the two approaches, indicating that while model-based RL agents may achieve quicker initial learning, model-free RL agents can ultimately reach comparable performance outcomes.

- Training Resources: Analysis of training resource requirements revealed that model-based

RL agents required approximately three times more weight updates and had networks with about three times as many weights compared to model-free RL agents. This increased computational demand highlights a significant consideration for applications where computational resources are limited or where rapid training is crucial. During this project this was especially apparent when working with the Atari:Freeway environment, due to computational and time limitations the model-based agent was not able to finish training successfully. This was both due to the relatively large demand for memory capacity and the increased demand for training time.

In conclusion, this paper provides insights into the comparative advantages and trade-offs between model-based and model-free reinforcement learning approaches. Model-based RL agents demonstrate superior sample efficiency and initial performance due to their ability to leverage pre-trained transition models. However, the higher computational demands associated with model-based RL underscore the importance of considering resource constraints in practical implementations.

The convergence of final performance metrics suggests that specific application requirements should guide the choice between model-based and model-free RL approaches. For scenarios prioritizing sample efficiency, model-based RL may offer distinct advantages. Model-based reinforcement learning however suffers from environments with complex dynamics since the amount of memory capacity needed to store and update the transition model and the computation cost to train the model directly scales with how complex the environment is. Conversely, model-free RL remains competitive in achieving comparable long-term performance outcomes while potentially requiring fewer computational resources. The key trade-off to consider is the difficulty and cost of obtaining samples from the environment. If samples are not particularly hard or expensive to collect, model-free RL is often the superior choice because training time becomes a major factor. Although model-free RL may incur higher costs in collecting samples, it benefits from significantly lower computational costs during training, which can greatly reduce the overall training time for the agent.

Future research could focus on optimizing model-based RL methods to reduce computational costs associated with training model-based RL agents this way model-based RL agents can be scaled to applications associated with complex environments. Due to time constraints, the reward thresholds in this project were determined arbitrarily. The reward threshold used to determine sample efficiency should not be arbitrarily chosen, as it was in this project. Instead, the threshold choices should be based on thorough reasoning and justification. One effective method is to use human performance as a benchmark. By analyzing the scores or rewards that proficient human players typically achieve, the reward threshold can be set relative to this benchmark.

References

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2012, 7). The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, *47*, 253–279. Retrieved from `http://arxiv.org/abs/1207.4708` `http://dx.doi.org/10.1613/jair.3912` doi: 10.1613/jair.3912

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Openai, W. Z. (2016, 6). OpenAI Gym. Retrieved from `https://arxiv.org/abs/1606.01540v1`

Kaiser, , Babaeizadeh, M., Miłos, P., Zej Osí Nski, B., Campbell, R. H., Czechowski, K., ... Ai, D. (2020). MODEL BASED REINFORCEMENT LEARNING FOR ATARI. Retrieved from `https://goo.gl/itykP8`

Lin, L.-J. (1992). Self-Improving Reactive Agents Based On Reinforcement Learning, Planning and Teaching. , *8*, 293–321.

Oh, J., Guo, X., Lee, H., Lewis, R., & Singh, S. (2015, 7). Action-Conditional Video Prediction using Deep Networks in Atari Games. *Advances in Neural Information Processing Systems*, *2015-January*, 2863–2871. Retrieved from `https://arxiv.org/abs/1507.08750v2`

Sutton, R. S. (1991, 7). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, *2*(4), 160–163. Retrieved from https://dl.acm.org/doi/10.1145/122344.122377 doi: 10.1145/122344.122377

Van Hasselt, H., Guez, A., & Silver, D. (2015, 9). Deep Reinforcement Learning with Double Q-learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2094–2100. Retrieved from https://arxiv.org/abs/1509.06461v3 doi: 10.1609/aaai.v30i1.10295

Watkins, C. J. C. H., & Dayan, P. (1992). Technical Note Q,-Learning. , *8*, 279–292.

A Hyperparameters

Table A.1: Hyperparameters

Reinforcement learning environments				
Hyperparameters	Cartpole	Acrobot	Lunar Lander	Freeway
Training duration	30000	30000	90000	500000
Learning rate	0.0001	0.0001	0.0001	0.0001
Minimum buffer size	1000	1000	1000	1000
Maximum buffer size	9000	9000	9000	9000
Discount factor	0.99	0.99	0.99	0.99
Batch size	128	128	128	128
$\hat{Q}_{target}$ Update frequency	37	37	37	37
$\hat{Q}$ Update frequency	16	16	16	16
Minimum ϵ	0.7	0.7	0.7	0.7
Maximum ϵ	1	1	1	1
ϵ decay factor	0.2	0.2	0.2	0.2
Reward threshold	350	-60	250	18
Planning steps	32	32	32	64

B Architecture Details

Table B.1: Transition model

Reinforcement learning environments				
Network architectures	Cartpole	Acrobot	Lunar Lander	Freeway
Input Dimension	5	7	9	33601
Layer 1	512	256	256	16800
Layer 2	256	512	512	8400
Layer 3	256	256	256	8400
Layer 4	N.A.	256	256	16800
Output Dimension	5	7	9	33601

Table B.2: Double Deep Q-network (DDQN) MultiLayer Perceptron (MLP)

Reinforcement learning environments			
Network architectures	Cartpole	Acrobot	Lunar Lander
Input Dimension	4	6	8
Layer 1	512	512	512
Layer 2	256	256	256
Layer 3	256	256	256
Output Dimension	2	3	4

Table B.3: Double Deep Q-network (DDQN) Convolutional Neural Network (CNN)

Reinforcement learning environments	
Network architectures	Freeway
Input Image	210 x 160 (greyscale)
Convolutional Layer 1	1 x 32, kernel size = 4, stride = 2
Convolutional Layer 2	32 x 64, kernel size = 4, stride = 2
Convolutional Layer 3	64 x 64, kernel size = 3, stride = 1)
Flatten layer	
Fully Connected Layer 1 input dimension	23552
Fully Connected Layer 2 input dimension	512
Output Dimension	3

C Cartpole

This appendix contains the figures with the full set of collected data on the Cartpole environment as discussed in Section 4.

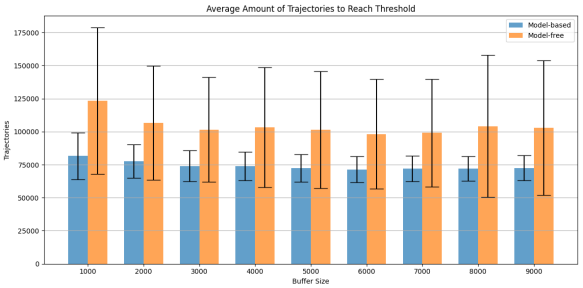
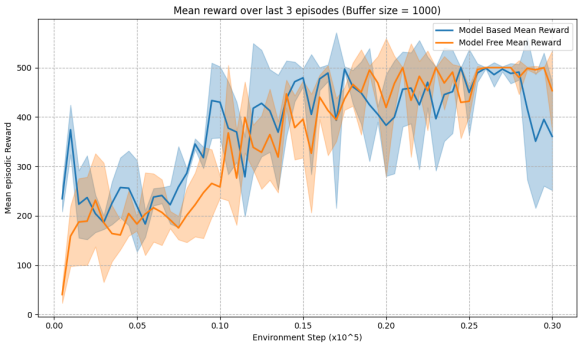
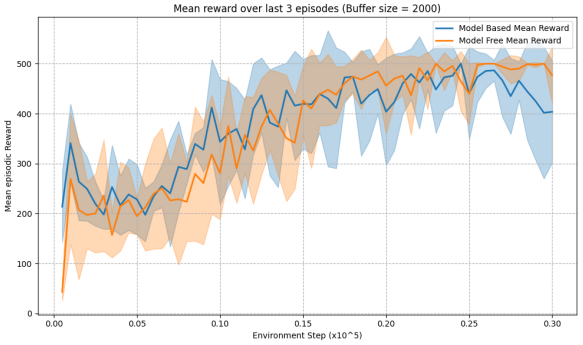
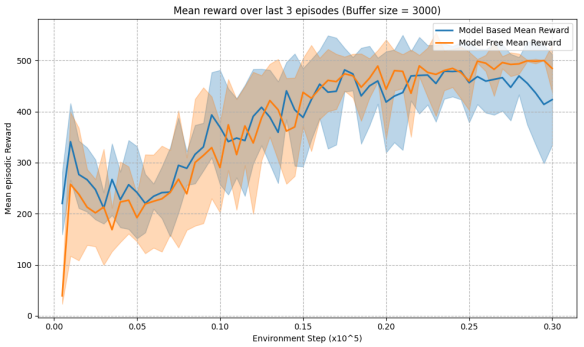
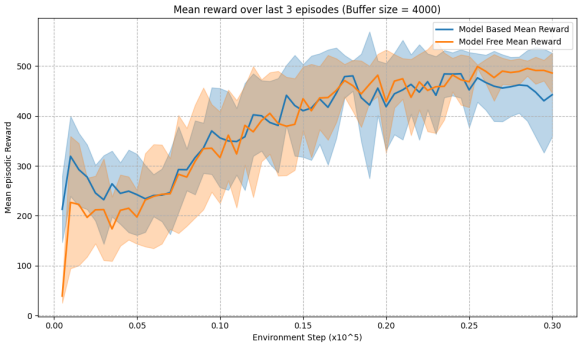


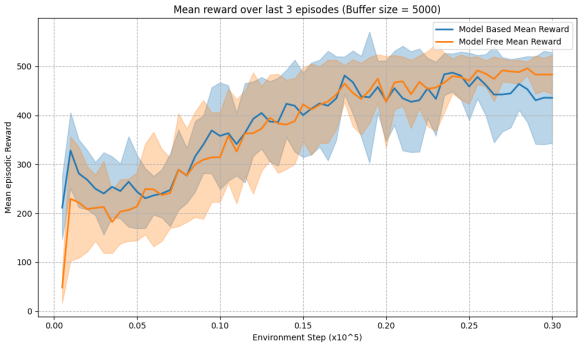
Figure C.1: Comparison of the amount of samples needed to reach the reward threshold for the Model-based and Model-free agents across buffer sizes

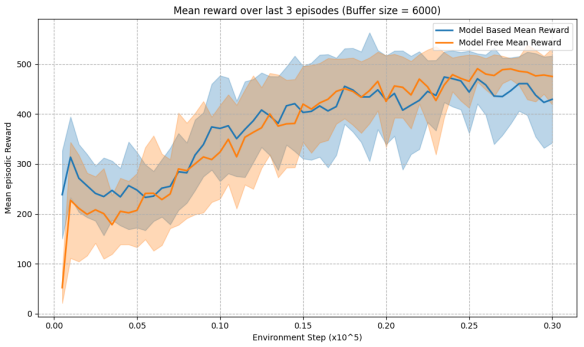












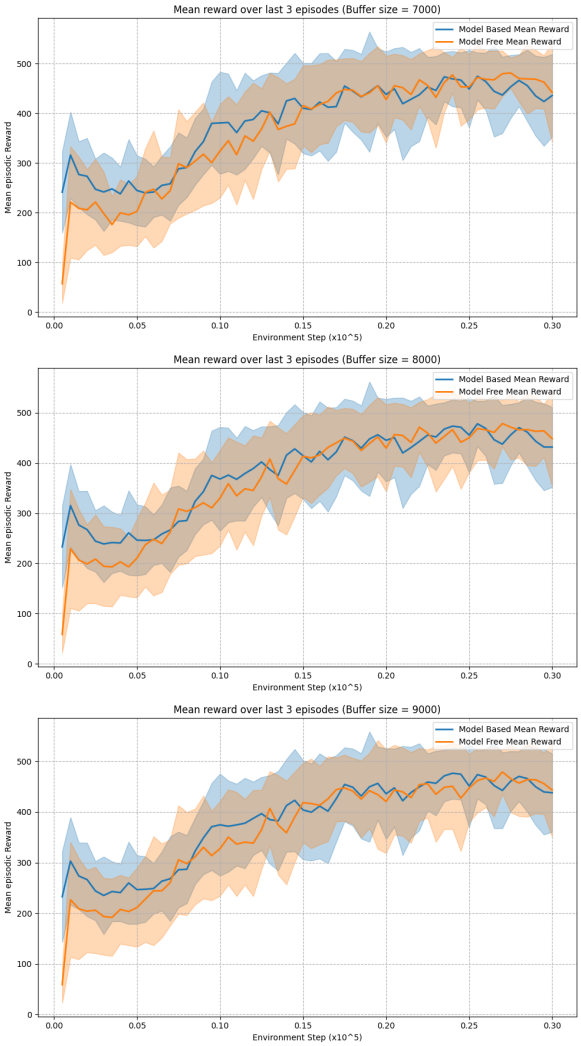


Figure C.2: Comparison of the Model-based and Model-free agent's mean reward over 3 episodes every 500 update steps across buffer sizes.

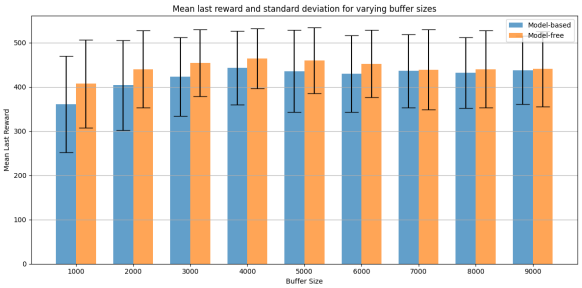


Figure C.3: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

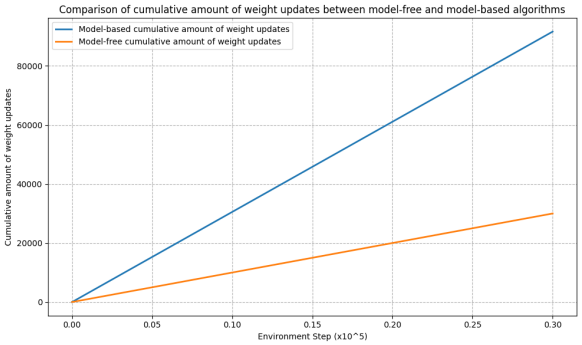


Figure C.4: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

D Acrobot

This appendix contains the figures with all of the collected data on the Acrobot environment. When looking at the number of trajectories to reach the reward threshold, we can see that model-based reinforcement learning is not only able to reach the threshold quicker but also more consistently across buffer sizes as can be seen from the smaller standard deviations. Similar to what was observed for the Cartpole environment the training process for both agents became more stable after increasing the buffer size. The jump-start effect that was discussed in Section 4.2 is also observed. The mean last reward of the agents stayed relatively stable for most buffer sizes. The large standard deviation observed for a buffer size of 1000 can be attributed to the stochastic nature of the environment. This is consistent with the conclusions drawn from the Cartpole data. Unsurprisingly the amount of weight updates of the Model-based agent was a multiple of the model-free weight updates, which is once again in line with the conclusions drawn previously.

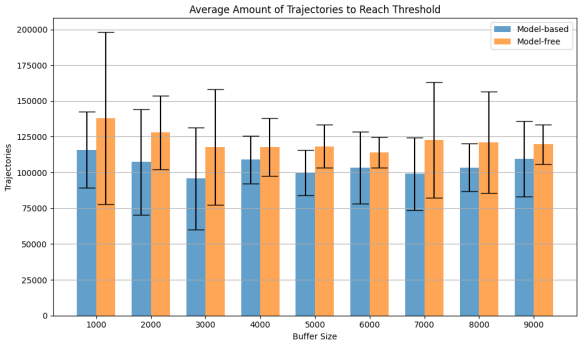
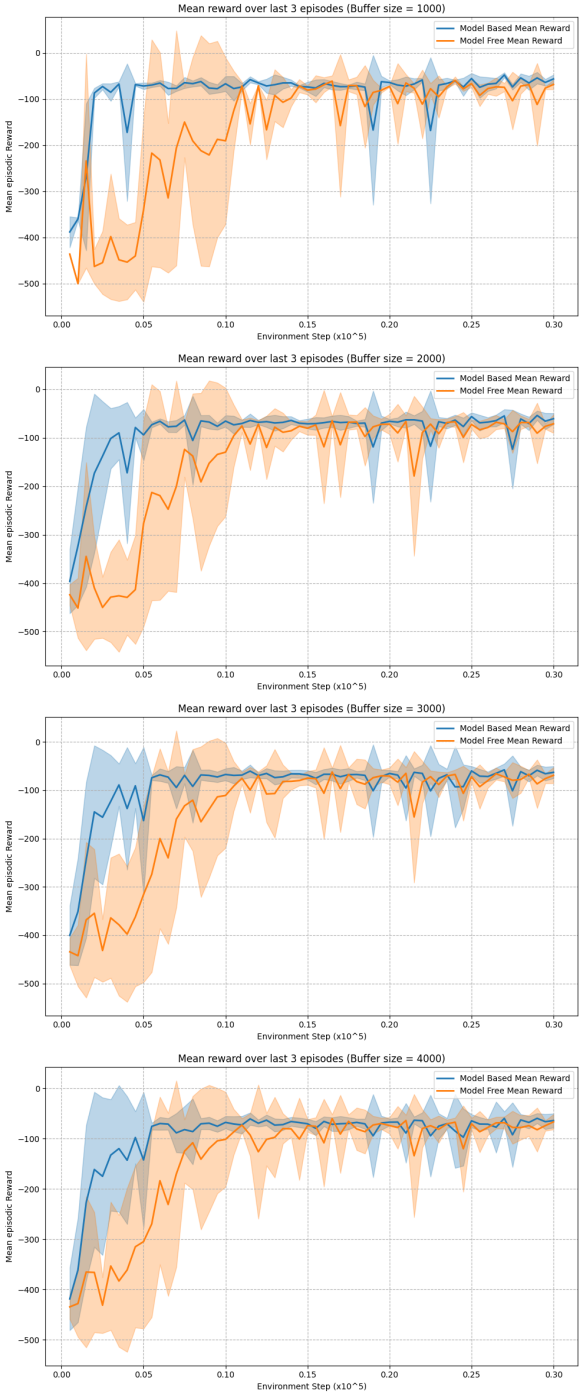
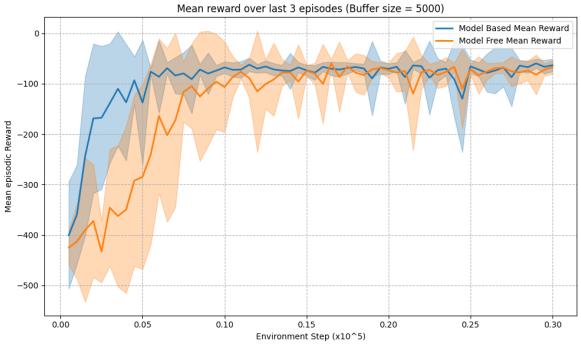
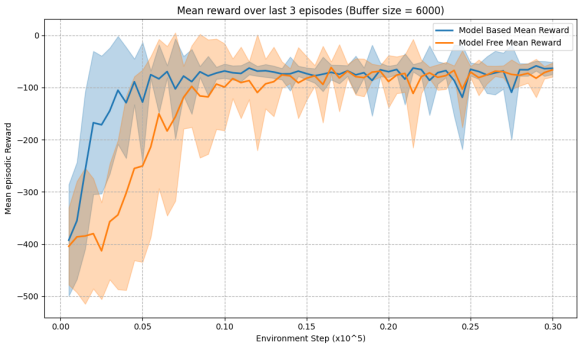
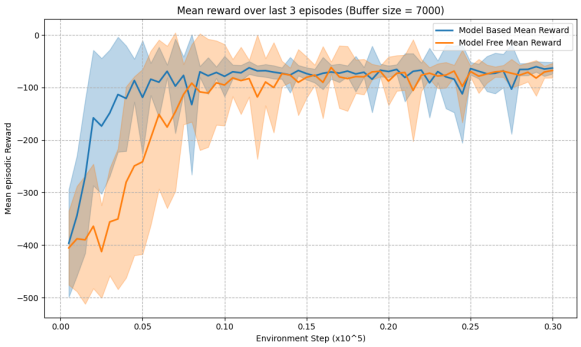


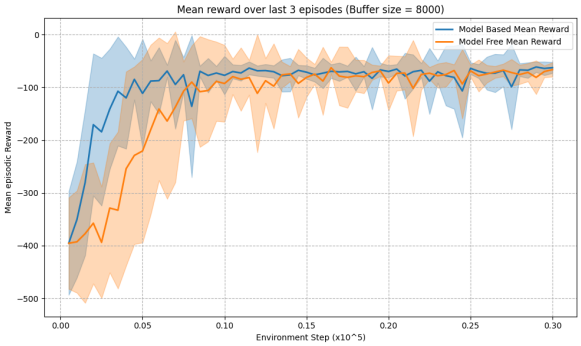
Figure D.1: Comparison of the amount of samples needed to reach the reward threshold for the Model-based and Model-free agents across buffer sizes











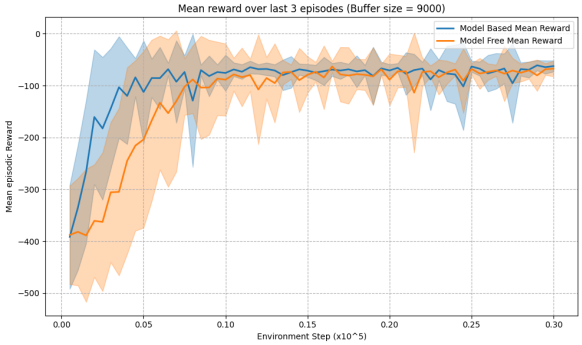


Figure D.2: Comparison of the Model-based and Model-free agent's mean reward over 3 episodes every 500 update steps across buffer sizes.

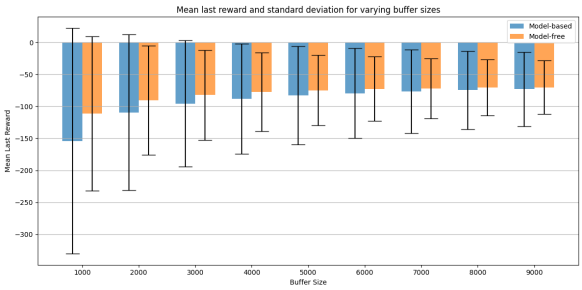


Figure D.3: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

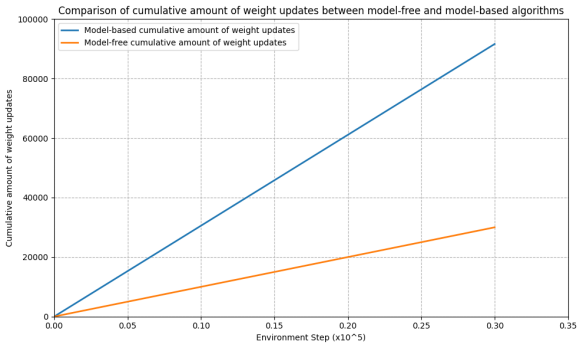


Figure D.4: Comparison between the Model-based and Model-free agent's amount of weight updates.

E Lunar Lander

This appendix contains the figures with all of the collected data on the Lunar Lander environment. When looking at average amount of trajectories to reach the reward threshold, we can once again see that model-based reinforcement learning consistently uses fewer trajectories to reach the threshold. Similar to what was observed for the Cartpole and Acrobot environments the training process became more stable for a higher buffer size and the jump-start effect was also observed. The mean last reward of the agents remained stable across buffer sizes, which is consistent with the conclusions drawn from the other environments. Finally, the amount of weight updates of the Model-based agent was more than twice as much as the model-free agent's weight updates.

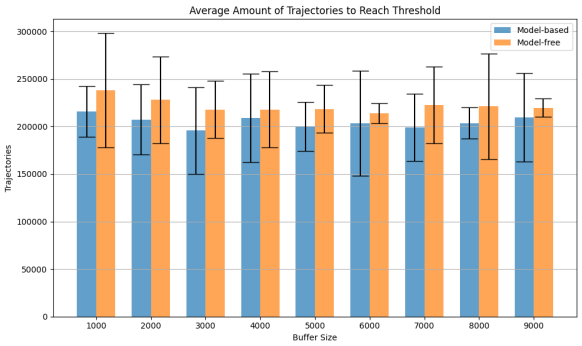
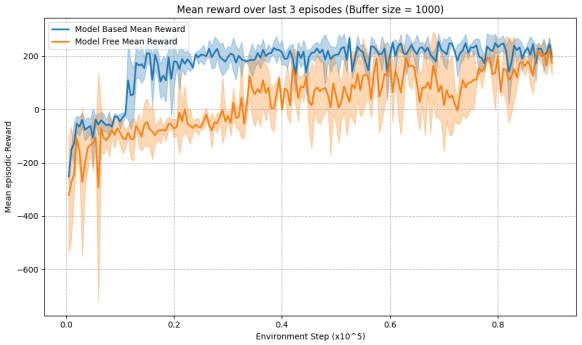
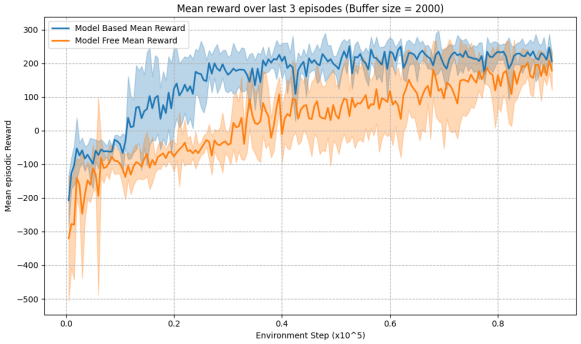
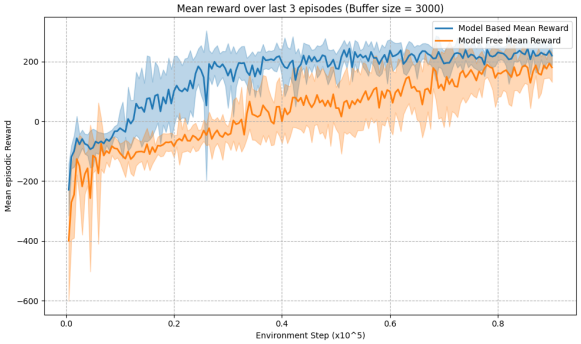
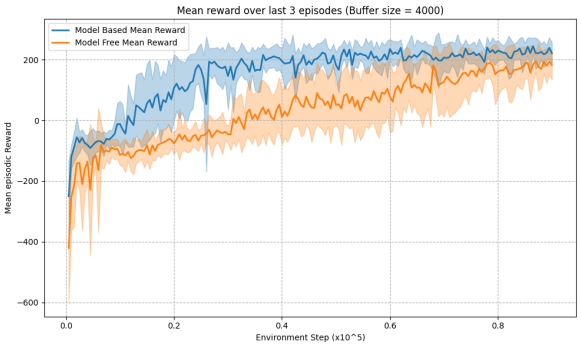


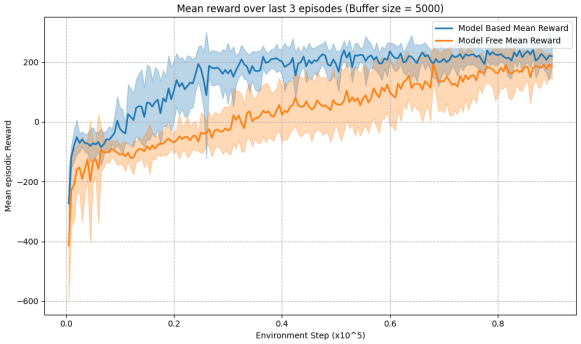
Figure E.1: Comparison of the amount of samples needed to reach the reward threshold for the Model-based and Model-free agents across buffer sizes

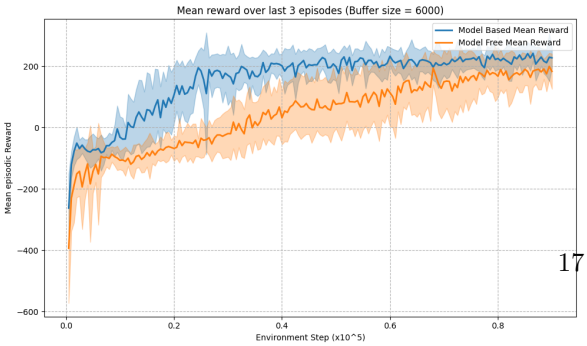












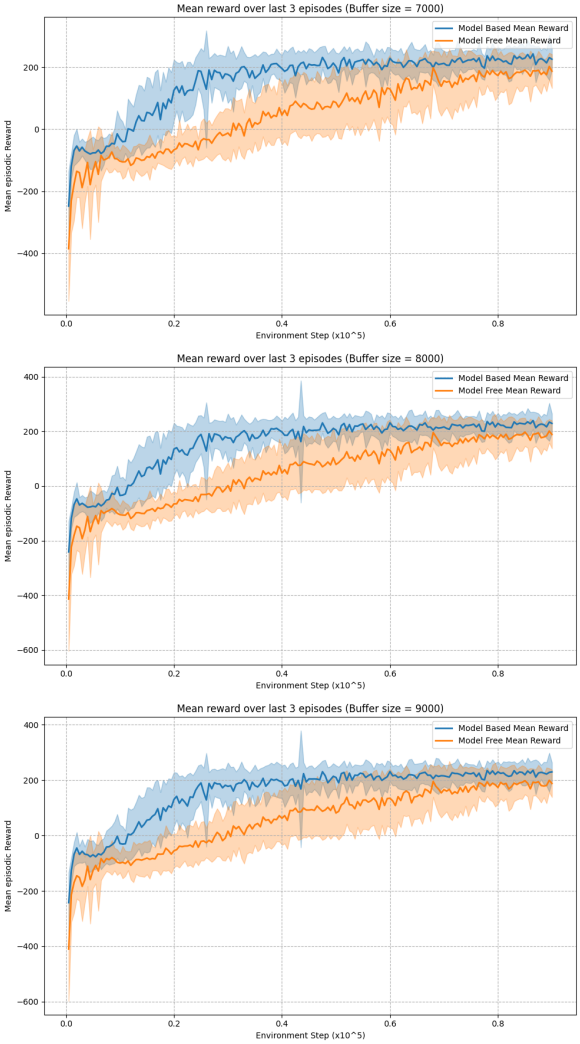


Figure E.2: Comparison of the Model-based and Model-free agent's mean reward over 3 episodes every 500 update steps across buffer sizes.

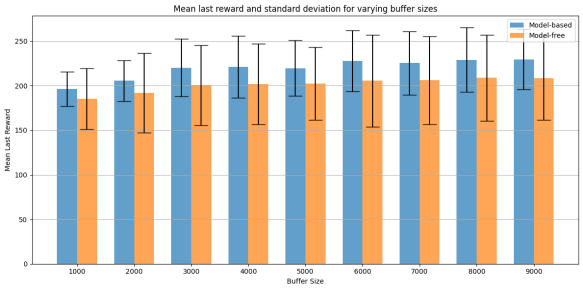


Figure E.3: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

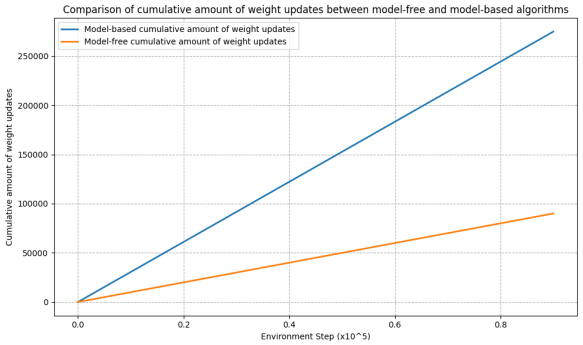


Figure E.4: Comparison between the Model-based and Model-free agent's amount of weight updates.

F Freeway

This appendix contains the figures with all of the collected data on the Atari:Freeway environment. Due to a constraint in computational resources and a limited amount of time to complete this project, the model-based agent was not able to be trained. Furthermore, the model-free agent was only trained on a buffer size of 9000. This also means that the data collected by the model-free agent, could not be compared. This again shows one of the major limitations of model-based reinforcement learning currently, high computational costs, because a transition model needs to be pre-trained and for every training iteration after both the weights of the transition model and the network of the agent need to be updated.

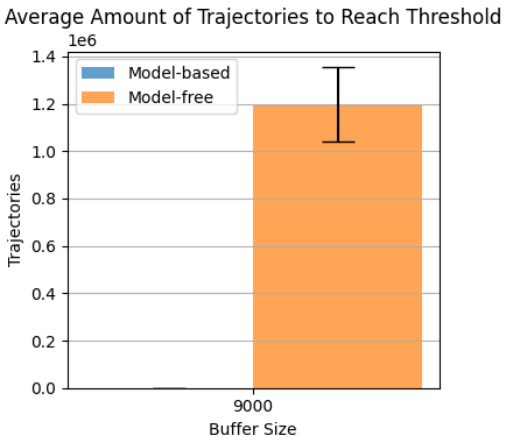


Figure F.1: Comparison of the amount of samples needed to reach the reward threshold for the Model-based and Model-free agents across buffer sizes

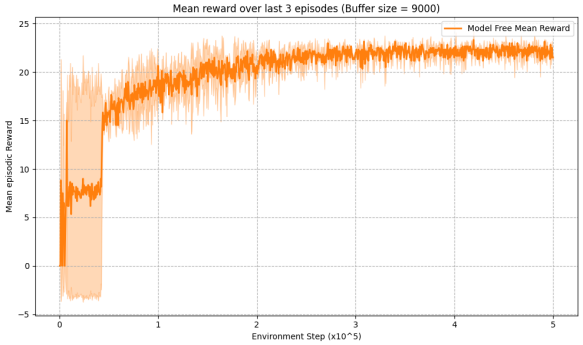


Figure F.2: Comparison of the Model-based and Model-free agent's mean reward over 3 episodes every 500 update steps across buffer sizes.

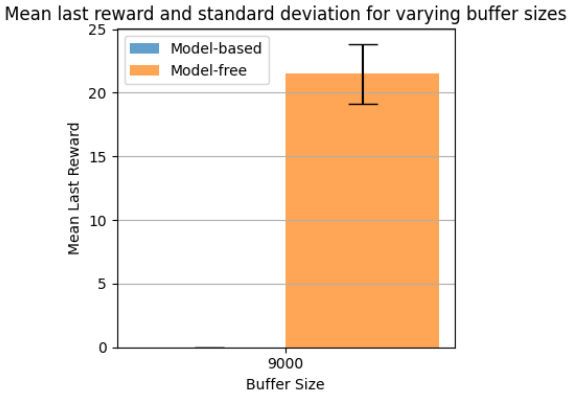


Figure F.3: Comparison between the Model-based and Model-free agent's mean last reward across buffer sizes.

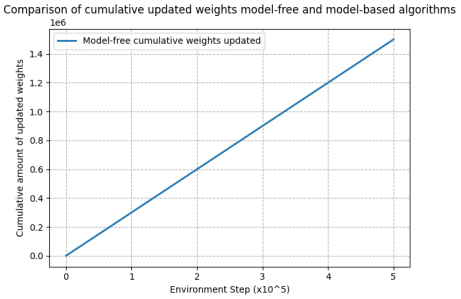


Figure F.4: Comparison between the Model-based and Model-free agent's amount of weight updates.