WESTERN SYDNEY
UNIVERSITY

Fast and incremental model-based reinforcement learning

Kin Man Lam
15823898

A report submitted for INFO7016 Postgradute Project A and INFO7017 Progradute Project B in partial fulfilment of the requirements for the degree of Master of Data Science

Supervisor: Dr. Oliver Obst

School of Computer, Data and Mathematical Sciences, Western Sydney University

Autumn 2023

Abstract

The aim of this report is to investigate possible acceleration of exploration in reinforcement learning using small neural networks that can efficiently learn from small data sets. The study uses of the Mountain Car Continuous environment from Gym and the Soft Actor Critic (SAC) from Stable Baselines3 (SB3) as the reinforcement learning algorithm. Multilayer perceptron (MLP), a type of feed-forward artificial neural network (ANN), is pre-built into SB3. Six MLPs of varying sizes were used for training and testing in the initial part of this report. Performance evaluations for these networks were basd on average rewards and the standard deviation of rewards over a number of episodes. The latter part of the report explores into fine-tuning and the application of Recurrent Neural Networks (RNNs) with behaviour cloning from the MLPs. The study results suggest that the use of SAC with medium-sized MLPs delivers the best performance among all the models explored in this report. As a result, it is recommended to maintain simplicity by adhering to the initial method.

Acknowledgements

I would like to express my special thanks and gratitude to my supervisor, Dr. Oliver Obst, who has been a key factor in enabling me to accomplish this report. He has provided me with continuous support and guided me in the right direction with patience throughout this research. Without his valuable suggestions and feedback, I would probably not have been able to complete this research. It is an honour to have such a knowledgeable, respectful, and friendly supervisor during the final stage of my study.

I would also like to thank Dr. Jianhua Yang for his support and commitment to me during this research. His workshops at the beginning helped me understand what research is, which was particularly helpful for students like myself who had no prior research experience.

Furthermore, I sincerely appreciate Western Sydney University for providing me with the opportunity to engage in research for the first time in my life. It has not only allowed me to grow as a researcher but has also helped me understand how research can be beneficial to humanity, especially for future generations.

Last but not least, I would like to express my deepest gratitude to my family for their continuous support and understanding during my difficult times. I truly wish that my recently deceased father would be proud of the research I have conducted in heaven.

Table of Contents

List of Tables

List of Figures

1 Introduction

The exploration-exploitation trade-off poses a significant challenge for agents in the field of reinforcement learning. Exploration and exploitation are two types of very different activity that a simulated agent can engage in environment. By definition, exploration focuses on gaining long term benefit by allowing the agent to improve its knowledge whereas exploitation allows the agent to use greedy approach to gain more immediate reward based on current estimated value [Rocca, 2021]. While modern reinforcement learning algorithms that optimise for the best returns can achieve good exploitation efficiently, exploration remains as an open topic [Sutton and Barto, 2018].

This research report will investigate the possibility of using small neural networks that can learn from small data sets to accelerate exploration in reinforcement learning. To achieve desirable results, tasks including but not limited to setting up an environment for training and testing simulated agents, collecting data from the environment to train small neural networks, using the neural network to predict possible future situations and learning policies via neural network as a model of the environment will be performed throughout the report. The underlying concept of the report is to examine methods that can learn an initial strategy from very short sequences and then potentially switch exploration strategies over time. As research in this area is limited, obtaining desirable results at the end of this report could potentially reduce time required to train and test agents in exploration and improve the model-based reinforcement learning overall.

2 Aims and Objectives

The main goal of the research in this report is to accelerate exploration in reinforcement learning through small neural networks that can learn from small data sets.

To achieve this objective, the report will address the following questions:

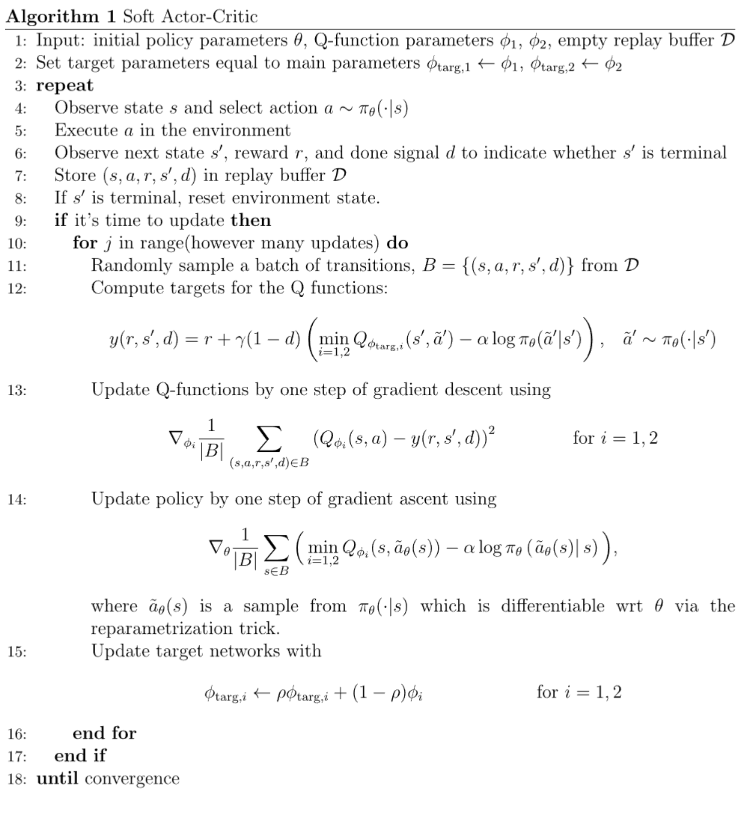
- Does the proposed approach which using multiple networks of different sizes effectively speed up exploration in reinforcement learning?
- If so, does the small neural networks show better performance when dealing with small data sets?

To answer the questions above, the details regarding the following items will need to be determined firstly in this report:

- The reinforcement learning algorithm used for the simulated agent
- The neural networks used for speed-up in exploration
- The environment in which the simulated agent operates

3 Literature Review

As stated previously, the purpose of this report is to investigate the use small neural networks that can learn from small data sets to speed up exploration in reinforcement learning. As a result, it is necessary to provide a briefly explanation of the reinforcement learning algorithm and the neural networks which will be used throughout the report.



```
Algorithm 1 Soft Actor-Critic
1: Input: initial policy parameters $\theta$, Q-function parameters $\phi_1$, $\phi_2$, empty replay buffer $\mathcal{D}$
2: Set target parameters equal to main parameters $\phi_{\text{targ},1} \leftarrow \phi_1$, $\phi_{\text{targ},2} \leftarrow \phi_2$
3: repeat
4:   Observe state $s$ and select action $a \sim \pi_\theta(\cdot|s)$
5:   Execute $a$ in the environment
6:   Observe next state $s'$, reward $r$, and done signal $d$ to indicate whether $s'$ is terminal
7:   Store $(s, a, r, s', d)$ in replay buffer $\mathcal{D}$
8:   If $s'$ is terminal, reset environment state.
9:   if it's time to update then
10:    for $j$ in range(however many updates) do
11:      Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from $\mathcal{D}$
12:      Compute targets for the Q functions:
           $y(r, s', d) = r + \gamma(1-d)\left(\min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s')\right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$
13:      Update Q-functions by one step of gradient descent using
           $\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d)\in B} \left(Q_{\phi_i}(s,a) - y(r,s',d)\right)^2 \qquad \text{for } i = 1, 2$
14:      Update policy by one step of gradient ascent using
           $\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \Big(\min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta\left(\tilde{a}_\theta(s)|\, s\right)\Big),$
         where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt $\theta$ via the
         reparametrization trick.
15:      Update target networks with
           $\phi_{\text{targ},i} \leftarrow \rho\phi_{\text{targ},i} + (1-\rho)\phi_i \qquad \text{for } i = 1, 2$
16:    end for
17:  end if
18: until convergence
```

Figure 1: The pseudocode for Soft Actor-Critic

Soft Actor-Critic (SAC) is an off-policy reinforcement learning algorithm which is designed for continuous actions. It is based on a maximum entropy reinforcement learning where the objective is to find the optimal policy that maximises the expected long term reward and long term entropy. SAC is a modified version of actor-critic algorithm which combines value-based and policy-based approach to reinforcement learning. High entropy can encourage exploration and avoids premature convergence to sub-optimal policies for the simulated agent [V.Kumar, 2019]. There are three networks employed in SAC, which are a state-value function V parameterised by ψ, a soft Q-function Q parameterised by θ, and a policy function π parameterised by ϕ. Moreover, SAC shows excellent sample efficiency and robustness to hyper-parameters. Figure 1 above shows the pseudocode for SAC [Haarnoja et al., 2019].

An artificial neural network (ANN) is an algorithm based on brain function, used to model complicated patterns and forecast problems. The ANN is a machine learning model which was inspired by the concept of biological neurons within the human brains. The nodes and their interconnections form the most important components of an ANN, emulating the role of synapses within a biological brain. Other essential elements include the input layer, hidden layer, output layer, weights and bias, activation function, back-propagation, and gradient descent [Singh, 2021]. Because of their ability to learn from and make decisions based on data, ANNs are widely used in image processing, character recognition, and forecasting in machine learning.

A recurrent neural network (RNN) is a type of neural network architecture that specifically adapted to work for time series data, natural language processing data or any data that involves sequences. RNNs have the concept of memory which helps them to store states or information from previous inputs to generate the next output of the sequence. It is possible to be achieved as the feedback loop in RNNs allow information to be passed within a layer in contrast to feed-forward neural networks in which information is only passed between layers [Dancker, 2022]. Bidirectional Recurrent Neural Networks, Gated Recurrent Units and Long Short Term Memory are the three most common variants of RNN [Saeed, 2022].

Overall, the advantages of SAC clearly make it to be the most suitable algorithm for this report. In pursuit of desirable results, SAC will be combined with ANNs and RNNs to create an effective approach for exploring reinforcement learning challenges.

4 Methodology

4.1 Application of Mountain Car Continuous Environment

The first task in this report is to identify an ideal environment and implement a chosen reinforcement learning algorithm within it. The choice for environment is Mountain Car Continuous from Gym library developed by OpenAI. This environment is a deterministic Markov Decision Process (MDP) that consists of a car placed stochastically at the bottom of a sinusoidal valley, with the only possible actions being the accelerations that can be applied to the car in either direction. The goal of the MDP is to strategically accelerate the car to reach the goal state on top of the right hill [Lin, 2022]. The continuous action version of this environment is employed with its details shown below.

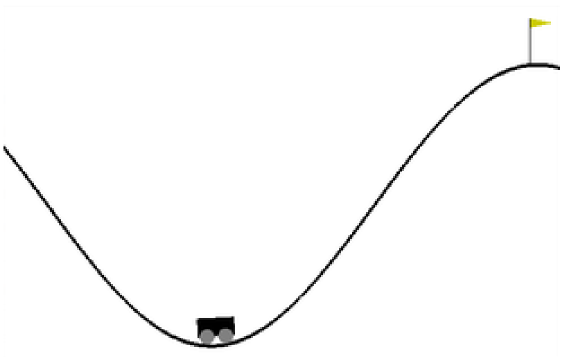
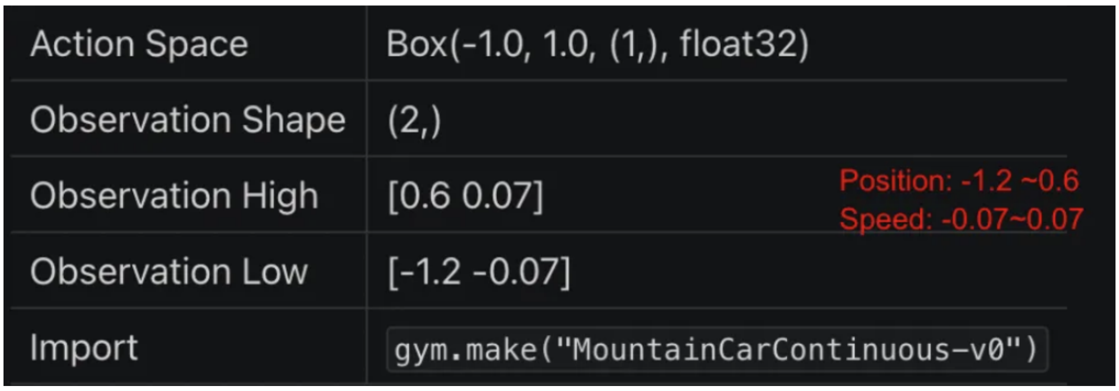


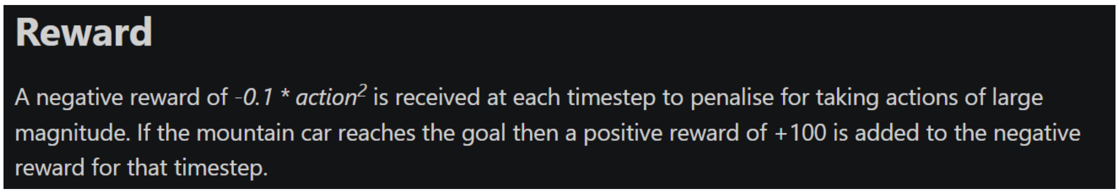
Figure 2: The view for Mountain Car Continuous environment



Action Space	Box(-1.0, 1.0, (1,), float32)
Observation Shape	(2,)
Observation High	[0.6 0.07]
Observation Low	[-1.2 -0.07]
Import	`gym.make("MountainCarContinuous-v0")`

Position: -1.2 ~0.6
Speed: -0.07~0.07

Figure 3: The parameters for Mountain Car Continuous environment


Reward

A negative reward of *-0.1 * action^2* is received at each timestep to penalise for taking actions of large magnitude. If the mountain car reaches the goal then a positive reward of +100 is added to the negative reward for that timestep.

Figure 4: The reward for Mountain Car Continuous environment

It is also worth to mention that the agent is penalised if it applies too much force or if it is far from the goal which affects the reward. Such penalties will encourage the agent to reach the top of the mountain with minimal effort, thus promoting more efficient exploration and learning [Lin, 2022].

4.2 Application of SAC and ANN Models in Stable Baselines3

Stable Baselines3 (SB3) is a set of reliable and high-quality reinforcement learning algorithms implemented in Python and built on top of the PyTorch deep learning framework. The library includes various algorithms such as A2C, DDPG, DQN, PPO and SAC [Raffin et al., 2021]. The SAC algorithm from the library has been chosen as reinforcement learning algorithm to be implemented into Mountain Car Continuous environment in this report. Multilayer Preceptron (MLP) is a type of feedforward ANNs integrated with the SB3 library. Therefore, multiple MLPs of different sizes will be trained and tested firstly in this report.

The initial step in the training and testing process is to create the environment and set up the parameters including the algorithm, policy network, size of networks and number of episodes. After training and testing the multiple networks of different sizes, benchmarks will then be established. All results will subsequently be recorded and compared in visual plots. If smaller networks can demonstrate higher average rewards and lower standard deviation, the research presented in this report could be considered as successful in its first stage.

4.3 Application of Fine-Tuning on Pre-Trained SAC and ANN Models

Transfer learning occurs when knowledge from a previously trained model is applied to a new, but related task. Although the source and target tasks in this study both involve the Mountain Car Continuous environment, this process is still a form of transfer learning and is often referred to as fine-tuning. Since the target task benefits from the learning achieved by the source task, there is a potential for improved final performance [Brownlee, 2017a]. Figure 5 shows an example of fine-tuning involves two ANNs.

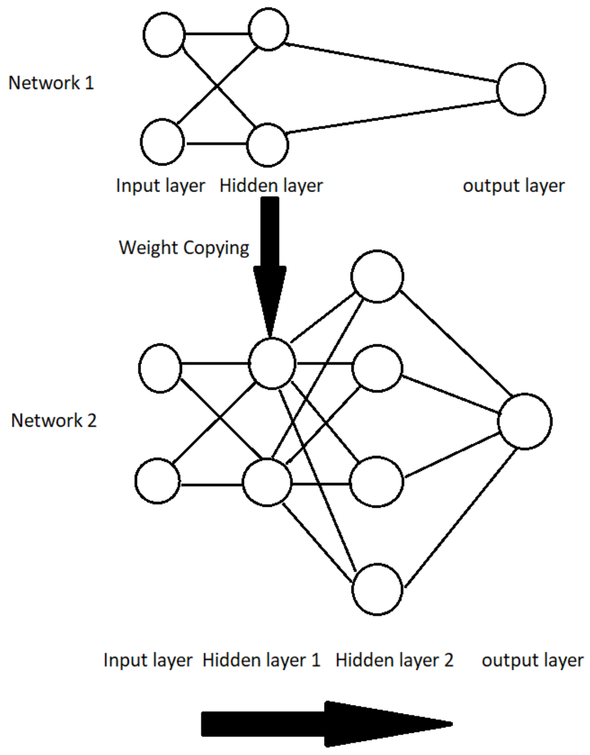


Figure 5: An example of fine-tuning involves two ANNs

The MLPs of varying sizes created in the earlier part of this report will be reused, with dividing them into pairs based on similarity of their network architectures in this part. For example, the first network has single hidden layer with 32 neurons, while the second network has double hidden layers with 32 neurons in the first one and 64 neurons in the second one. Fine-tuning is possible in this case since both networks have 32 neurons in their first hidden layers. Therefore, weights from the first network will be extracted and transferred to the second network, which will then continue its training [Baheti, 2021]. The final performance of the combined network will be examined to check for any improvements.

4.4 Application of RNNs using Behaviour Cloning on Pre-Trained SAC and ANN Models

The next stage of the report will involve using RNNs instead of MLPs as the neural network for training and testing models. Since SB3 does not have built-in support for RNN policies, a process known as behaviour cloning or imitation learning will be used. The concept involves training separate RNN models to mimic the behaviour of the pre-trained SAC models from the earlier stage [Lőrincz, 2019]. Figure 6 shows an example of behaviour cloning involves an ANN and an RNN.

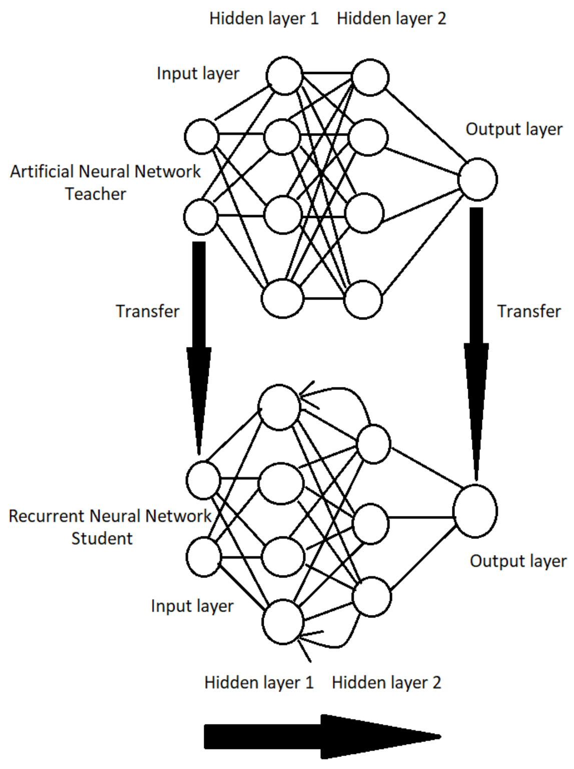


Figure 6: An example of behaviour cloning involves an ANN and an RNN

For the Mountain Car Continuous environment, the two elements of the observation which are position and velocity will be used as inputs. On the other hand, the output will be the action. During the training process, the RNN models will be given a sequence of observations from the pre-trained SAC models as inputs. The RNN will use these inputs to generate outputs, which are actions.

Then a loss function will compare the predicted action to the actual action from the pre-trained SAC models and calculates the difference between them. The parameters of the RNN models will be updated to minimise this difference afterwards. This process represents a type of supervised learning. In other words, the SAC models can be seen as teachers providing guidelines for the students which are the RNN models during their training process. As a result, the RNN models will learn how to generate the correct outputs and may potentially improve the actions they generate after training by adjusting their parameters [Torabi et al., 2018].

Subsequently, multiple RNNs of varying sizes will be tested and compared to evaluate improvements over the pre-trained SAC models and to determine if a smaller network size gives better performance.

4.5 Expected Outcomes

The performances for all models trained using the methods above will be shown and compared in the next part of this report. As a result, this will help to determine the best method for Mountain Car Continuous environment. If a smaller size neural network shows better performance with that particular method, it will prove fast and incremental model-based reinforcement learning in exploration can be achieved with smaller neural network and small data sets. Therefore, this highly efficient and effective method for exploration in reinforcement learning could be applied in future use cases.

5 Research Results and Analysis

5.1 Evaluation of Original Models' Performance

Table 1: The MLPs used for the original models

Network	Size of actor network	Size of critic network
1 (one layer)	8	8
2 (one layer)	16	16
3 (two layers)	32, 32	32, 32
4 (two layers)	64, 64	64, 64
5 (two layers)	128, 128	128, 128
6 (three layers)	256, 256, 256	256, 256, 256

The MLPs shown in the table above were trained and tested with SAC in Mountain Car Continuous environment. The network size increased in descending order. It is necessary to point out that network 4 was the default size for MLP policy in SB3. To illustrate the agent's progress, Figure 7 shows the training rewards over 50,000 timesteps for all networks. The results suggest that rewards approach zero at around the 25,000 step-mark and do not advance beyond that point except for network 2. Network 2 was initially running behind but eventually caught up with the others at around the 41,000 step-mark. Therefore, a training process of 50,000 timesteps was deemed appropriate in this case. The training process, which was relatively lengthy, took approximately one hour.

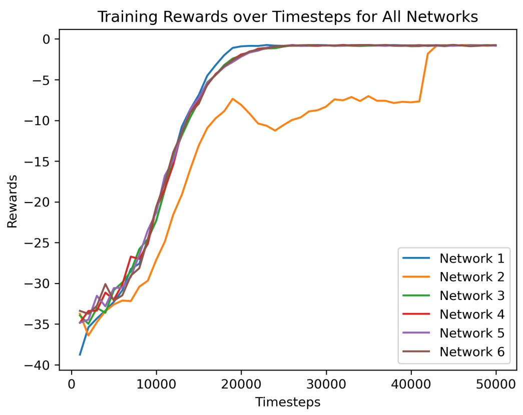


Figure 7: The training rewards over timesteps for the original models

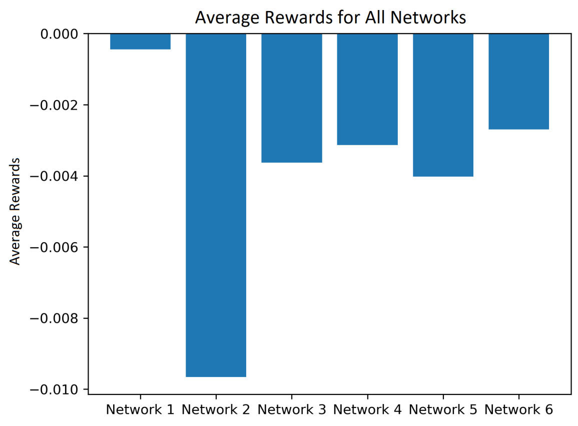


Figure 8: The average rewards for the original models

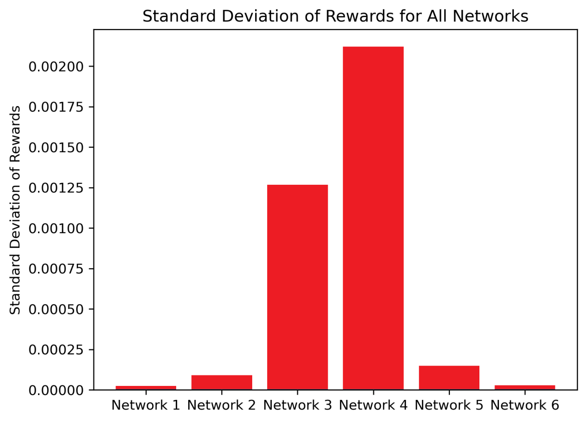


Figure 9: The standard deviation of rewards for the original models

To evaluate the performance of the networks, average rewards and standard deviation of rewards over 100 episodes were used as measurements. Figure 8 shows the average rewards for all networks where a higher number represents better performance. On the other hand, Figure 9 shows the standard deviation of rewards for each network with a lower number indicating more consistency and reliability for the network [Gupta, 2020].

The results in Figure 8 suggests network 1 has the highest number in average rewards over other networks. Additionally, Figure 9 shows both network 1 and network 6 have the lowest standard deviation of rewards among all networks. As a result, it indicates the smallest network delivers the best performance in both measurements closely followed by the largest network. Nevertheless, more training and testing with other methods are still necessary before any conclusion can be drawn.

5.2 Evaluation of Modified Original Models' Performance

Table 2: The MLPs used for the modified original models

Network	Size of actor network	Size of critic network
1 (one layer)	32	32
2 (one layer)	64	64
3 (two layers)	32, 64	32, 64
4 (two layers)	64, 128	64, 128
5 (three layers)	32, 64, 128	32, 64, 128
6 (three layers)	64, 128, 256	64, 128, 256

The MLPs above were modified from their original models to achieve more consistency during training and these modifications can be reused in later parts of this report. The first two networks in the original models which had a single layer and a small amount of neurons were unstable as the range of their average rewards varied significantly between different training sessions. Besides, the modified models enabled fine-tuning which will be discussed in the next part. The number of timesteps was reduced in this case since the learning curves for all networks perform similarly and they approach zero in the 25,000 step-marks as shown in Figure 10. Therefore, the duration of the training process was also cut down to approximately half an hour.

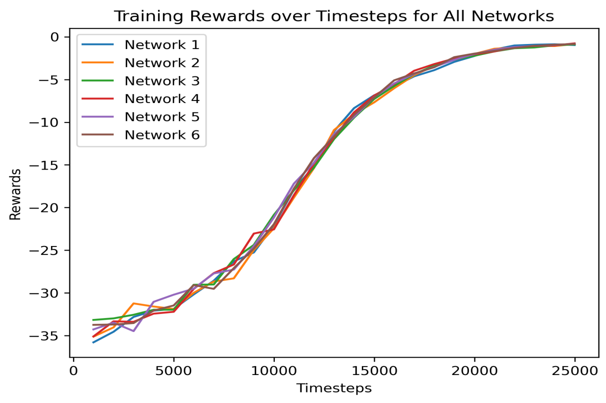


Figure 10: The training rewards over timesteps for the modified original models

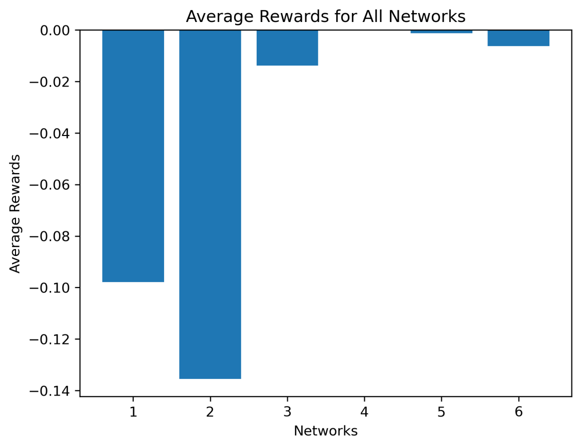


Figure 11: The average rewards for the modified original models

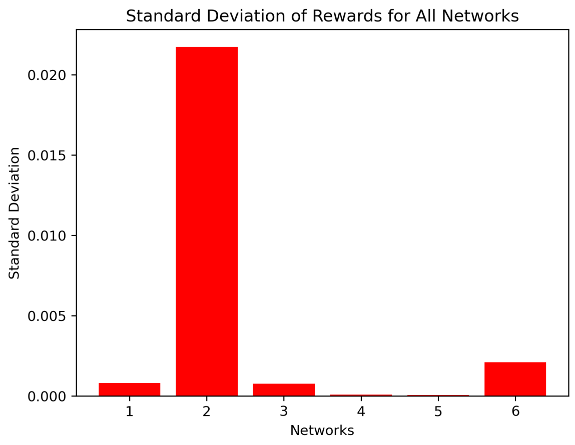


Figure 12: The standard deviation of rewards for the modified original models

The average rewards and standard deviation of rewards for all networks were also calculated over 100 episodes for evaluation. Nevertheless, the results showed notable differences compared to the original models.

Figure 11 suggests that network 4, which is the medium-sized network, has the best performance in terms of average rewards among all networks as the number is very close to zero. On the other hand, network 1 and 2 show more stability but perform worse than others which contrasts with the performance of the original models. As for standard deviation, Figure 12 shows both network 4 and 5 have the lowest numbers among all networks. These results suggest that medium-sized networks should be the preferred choice for SAC and ANN models when training in Mountain Car Continuous environment.

5.3 Evaluation of Fine-Tuning on Modified Original Models

Table 3: The MLPs used for fine-tuning on the modified original models

Network	Size of actor network	Size of critic network
1 (one layer)	32	32
3 (two layers)	32, 64	32, 64
3 (two layers)	32, 64	32, 64
5 (three layers)	32, 64, 128	32, 64, 128
2 (one layer)	64	64
4 (two layers)	64, 128	64, 128
4 (two layers)	64, 128	64, 128
6 (three layers)	64, 128, 256	64, 128, 256

The networks from the previous part were divided into four pairs as shown in Table 3, with the first model in each pair serving as a source for weight transfer to the second model. For instance, weights from the first network were transferred to the third network as the number of neurons in their first hidden layers is identical.

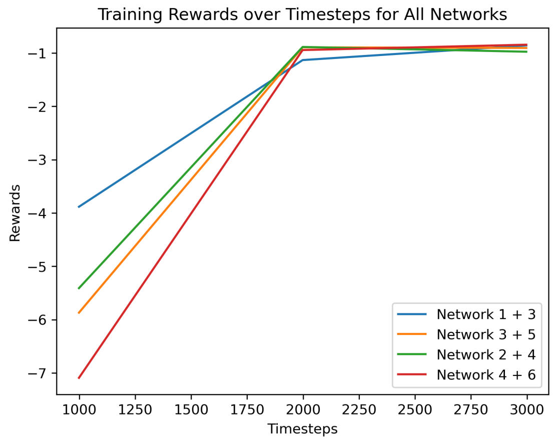


Figure 13: The training rewards over timesteps for the fine-tuned modified original models

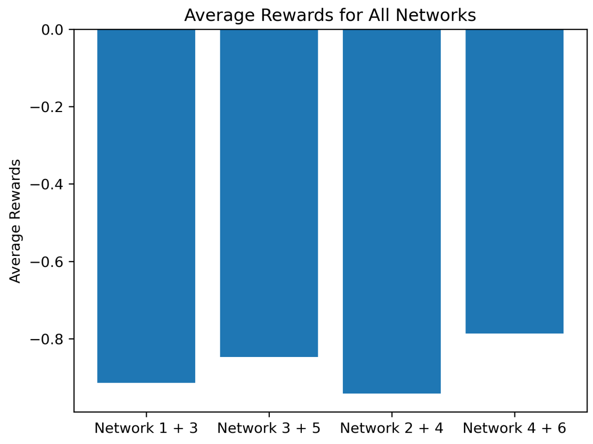


Figure 14: The average rewards for the fine-tuned modified original models

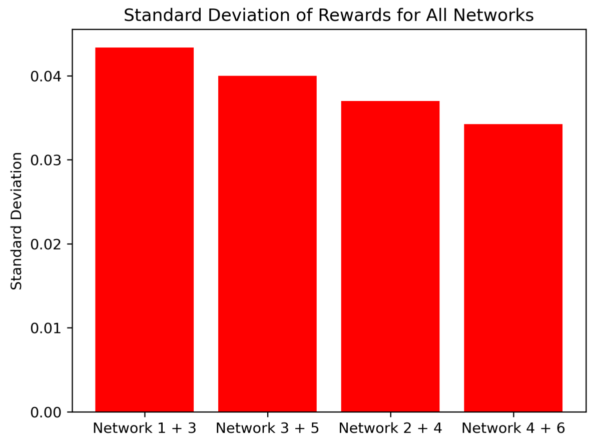


Figure 15: The standard deviation of rewards for the fine-tuned modified original models

The training process for fine-tuning was reduced to less than three minutes due to the pre-loading of the modified original models. All recipient models were fine-tuned over 3,000 timesteps after weights transfer. In addition, each model was evaluated over 100 episodes in the environment. Figure 13 indicates training rewards for all recipient models peaked at 2,000 timesteps. Figure 14 shows the combined network comprising network 4 and 6 had the highest average rewards while Figure 15 indicates this model had the lowest standard deviation. However, the performance for all the fine-tuned models was significantly poorer than that models discussed earlier both in terms of average rewards and standard deviation. Possible reasons for this subpar performance could include overfitting of the transferred weights, interference caused by the transfer or discrepancies in the architecture of the deeper layers in the combined network.

5.4 Evaluation of the Implementation of RNNs with Behaviour Cloning on Modified Original Models

The input and output for the RNNs are transferred from the modified original models discussed earlier. These RNNs use a Long Short Term Memory architecture with 64 neurons in their single hidden layer [Brownlee, 2017b]. For each of these modified original models, 10,000 steps were collected from the environment to create state-action pairs. These pairs were then converted into sequences of five to serve as the inputs and outputs for training the RNNs. Both Mean Squared Error and Adam optimiser were used during the training process. Similar to the previous process, the pre-loading of the modified original models allowed the training duration to be reduced to just one minute.

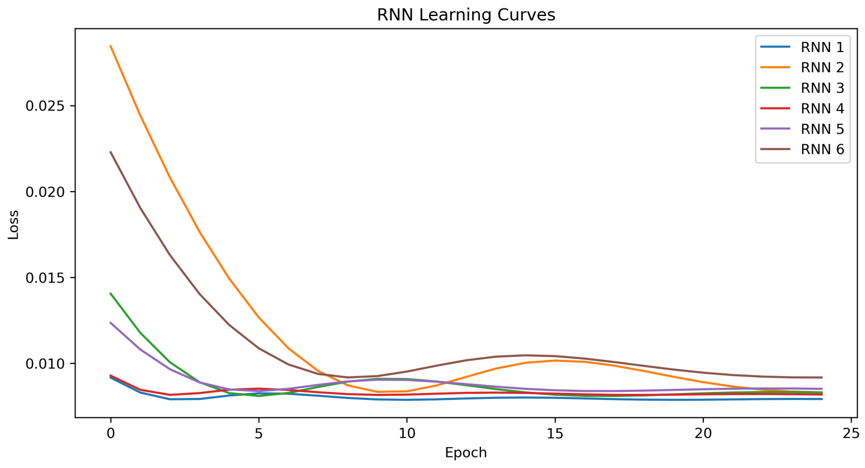


Figure 16: The learning curves for the RNN models

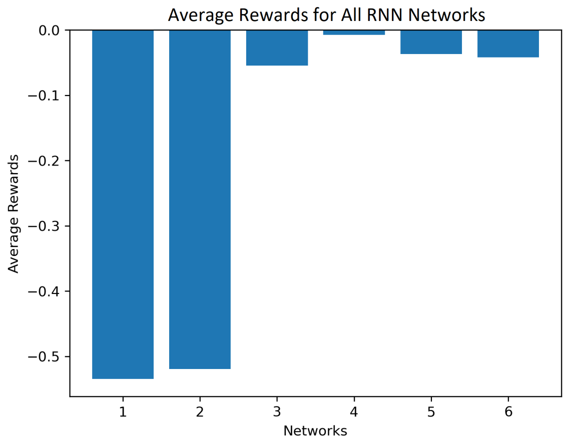


Figure 17: The average rewards for the RNN models

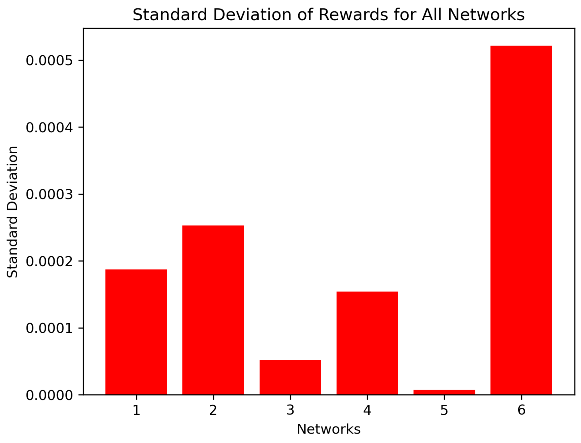


Figure 18: The standard deviation of rewards for the RNN models

A total of 100 episodes were for evaluation purposes. Figure 16 shows that the learning curves for all RNN models minimised the loss within 25 epochs. Meanwhile, Figure 17 shows network 4 had the highest average rewards and Figure 18 suggests network 5 had the lowest standard deviation among all RNN models. Although there are minor differences, their overall performance did not show a significant improvement over the modified original models.

6 Conclusions and Recommendations

In order to achieve speed up exploration in reinforcement learning with small neural networks for small data sets, this report uses the Mountain Car Continuous environment from Gym and the SAC algorithm from SB3. MLP, a type of feed-forward ANN, is the built-in neural network in SB3 and it does not require customisation of the network architecture for use. Initially, six MLPs of varying sizes were trained and tested. Moreover, fine-tuning was applied to the MLPs and RNNs were implemented with behavior cloning from the MLPs to investigate potential performance improvements.

Nevertheless, the fine-tuning method did not show satisfactory performance in evaluation. Furthermore, the RNNs displayed performance on par with the original MLPs but no significant improvement was observed. Given that Mountain Car Continuous is a relatively simple environment, the use of SAC and MLPs has already proven to be sufficient and capable of delivering impressive performance for reinforcement learning. Therefore, it is recommended to maintain simplicity. The use of SAC and the built-in medium-sized neural networks in SB3 could provide the optimal outcome for reinforcement learning in the Mountain Car Continuous environment.

References

Pragati Baheti. A complete guide to transfer learning, 2021. URL `https://www.v7labs.com/blog/transfer-learning-guide`. Last accessed: 22 May 2023.

Jason Brownlee. Transfer learning for deep learning, 2017a. URL `https://machinelearningmastery.com/transfer-learning-for-deep-learning/`. Last accessed 22 May 2023.

Jason Brownlee. A gentle introduction to long short-term memory networks for experts, 2017b. URL `https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/`. Last accessed 26 May 2023.

Jonte Dancker. A brief introduction to recurrent neural networks, 2022. URL `https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4`. Last accessed 03 April 2023.

Anamika Gupta. Use of standard deviation in machine learning, 2020. URL `https://devstudioonline.com/article/use-of-standard-deviation-in-machine-learning`. Last accessed 07 May 2023.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

Renee Lin. Use stable baselines3 to solve mountain car continuous in gym, 2022. URL `https://reneelin2019.medium.com/use-stable-baselines3-to-solve-mountain-car-continuous-in-gym-3216912cd5e3`. Last accessed 02 April 2023.

Zoltán Lőrincz. A brief overview of imitation learning, 2019. URL `https://smartlabai.medium.com/a-brief-overview-of-imitation-learning-8a8a75c44a9c`. Last accessed 24 May 2023.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL `http://jmlr.org/papers/v22/20-1364.html`.

Joseph Rocca. The exploration-exploitation trade-off: intuitions and strategies, 2021. URL `https://towardsdatascience.com/the-exploration-exploitation-dilemma-f5622fbe1e82`. Last accessed 02 April 2023.

Mehreen Saeed. An introduction to recurrent neural networks and the math that powers them, 2022. URL `https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/`. Last accessed 03 April 2023.

Gourav Singh. Introduction to artificial neural networks, 2021. URL `https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks/`. Last accessed 20 May 2023.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation, 2018.

Vaishak V.Kumar. Soft actor-critic demystified, 2019. URL `https://towardsdatascience.com/soft-actor-critic-demystified-b8427df61665`. Last accessed 03 April 2023.

Appendix: Computer Code

The Python code for the research report can be found at the following URL:
https://github.com/lamkinman/Project.git