Reversal Generative Reinforcement Learning

Po-Lung Wang[1]

[1]Taipei, Taiwan

June 13, 2025

Abstract

We introduce a simple method for enabling pure deep neural networks to engage in reinforcement learning. The approach treats state and action as input data and reward as output data. First, when neural nets observe state, the neural nets initialize and update future actions using error back-propagation, with desired rewards as target. Second, the agent (neural nets) executes the updated future actions and learns the ensuing future rewards. The above learning process is a continuous iterative process until the agent's learned parameters approximate the real environment.

This method is applicable across diverse digital environments. Furthermore, it circumvents the need for handcrafted functions such as the Bellman function for propagating future rewards to actions, further leveraging the flexibility and capacity of deep neural networks. We substantiate the approach with experimental results and provide GitHub repository[1] in this paper.

1 Introduction

Deep reinforcement learning (Deep Q-Learning) [14] is one of the most popular algorithms in reinforcement learning. By leveraging the flexibility of deep neural network and the reward mechanism derived from the Bellman equation, it is widely used in areas such as robotics, gaming, and autonomous systems.

Deep reinforcement learning utilizes a deep neural network to map state (input) to Q-values for each possible action (output). During training, the Bellman equation is used to propagate future rewards to these Q-values, helping the network learn which action is more valuable in the present state. Namely, the update rule for the Q-value of a state-action pair (s, a) is given by: $Q(s,a) \leftarrow Q(s,a) + \alpha\left(r + \gamma \cdot \max_{a'} Q(s',a') - Q(s,a)\right)$ where α is the learning rate and γ is the discount factor representing how much future rewards are considered (see also Volodymyr et al. [13]).

[1] See: https://github.com/Brownwang0426/RGRL.

After training, an agent can determine the optimal action by selecting the action with the highest Q-value forwarded from the current state. From this perspective, the agent's action is directly derived from the output of the deep neural network, which is a straightforward and immediate process.

While theoretically sound, the Bellman equation is based on a handcrafted formula that assumes a perfect model of the environment's dynamics (i.e., how present state and future actions affect future rewards and how future rewards are distributed to future actions). This assumption could be limiting in complex environments where modeling these dynamics explicitly is difficult, which more or less results in the fact that the choice of hyperparameter such as discount factor γ might notably affect the over performance of an agent (as shown by Henderson et al. [7]). Therefore, multiple works are proposed to introduce more flexibility.

For example, Policy Gradient Methods (by Sutton et al. [19]) were proposed to offer more flexibility compared to methods based on the Bellman equation. In these methods, the gradient is derived mathematically from the objective of maximizing the expected cumulative reward. However, the expected cumulative reward itself, along with the discount factor, relies on human expert design.

There is also work proposing RL frameworks that do not rely on the Bellman equation. For example, "Model-Free Episodic Control (MFEC)" by Blundell et al. [1] proposes an alternative to value function-based methods. MFEC utilizes a non-parametric memory (a table of previous experiences) of state-action-reward tuples, where each entry stores information from individual episodes. This allows the agent to make decisions based on direct look-ups of similar past experiences instead of relying on learned value functions.

There are also studies focusing on decision-making by agents solely through deep neural networks, without the use of explicit reward functions, discount factors, or look-up tables. Since the Bellman equation and other human-designed functions typically average, assign, or predict future rewards, these methods could be replaced by deep neural network, modeled as $f(s, a) = r$, where s and a represent the input state and action, and r denotes the output reward. For example, a work by Yize et al. [2] employs gradient-based optimization through backpropagation in a recurrent neural network to adjust a time sequence of input data (e.g., energy usage) to minimize overall energy consumption in a building. Also, Miguel et al. [18] utilize similar gradient method to optimize wireless network configurations. Wang [21] extends these approaches by borrowing concept from Deep Dream (Mordvintsev et al. [15]) and iteratively refining input actions a via forward passes and backpropagation while freezing the neural network weights before executing intput actions a. Additionally, the author introduces a approach called 'Multi-Weight-Matrices Stochastic Gradient Descent (MWM-SGD),' where an agent leverages an ensemble of trained neural networks during the process of input gradient descent in order to avoid numerical local minima[2],

[2] For clarity, we distinguish between two types of "local minima": "numerical local minima," which refer to suboptimal solutions in the parameter space of the neural network model, and "environmental local minima," which refer to suboptimal solutions within the environment.

thereby enhancing decision-making performance. However, the aforementioned decision-making works are conducted within a supervised learning framework, with limited exploration in reinforcement learning.

Aside from the previous works, there are also medical and neuroscientific studies that support the idea that human actions are not the result of spontaneous, instantaneous events in the brain but rather emerge from ongoing updates or gradual build-up of activity in certain brain areas, like the motor cortex, before an action is executed.

For example, Kornhuber et al. [10] founded Readiness Potentials (RPs), which are slow electrical brain signals that occur before intentional motor actions. RPs involve a gradual, slow increase in electrical activity that reflects the brain's preparatory processes leading up to the execution of an intentional action, such as lifting a hand or pressing a button (see also Deecke [4]).

Libet et al. [11] and Haggard et al. [6] highlight two distinct phases of RPs. In the early stage, the brain initiates general motor preparation. In the second stage of RPs, the brain fine-tunes the actual motor commands in the primary motor cortex before executing them.

Schurger et al. [17] also present a model in which neural activity builds up gradually prior to the initiation of intentional movements. It describes how motor actions are prepared and updated over time before execution.

Desmurget et al. [5] further discuss how motor control involves both efferent signals (motor command) and reafferent signals (motor feedback), illustrating how the brain uses feedback to adjust and refine motor actions preparation using both signals iteratively before motor execution (see also Wolpert et al. [22]).

From this perspective, Deep Q-Learning's use of a deep neural network is a good approach. However, its viewing actions as output of a neural network seems to conflict with the finding of the above medical and neuroscientific studies. Since, in Deep Q-Learning, after training or learning, the motor actions of an agent are the direct forward-feeding result of a deep neural net, it is more spontaneous and instantaneous than an ongoing update. Furthermore, it involves little iterative interaction between efferent signals (motor command signals) and reafferent signals (motor feedback signals) to fine-tune motor actions prior to execution.

Given the limitations of the Bellman function, and the missing alignment with biological decision-making processes, this paper seeks to explore an alternative and biological possible approach to bridge the gap between reinforcement learning models and phenomenon observed in decision-making neural systems.

2 Algorithm

We first notice that the brain has many pathways from later layers back to earlier ones, and it could use these pathways in many ways to convey the information required for learning or error-backpropagation (as in Hinton [8]). This suggests potential parallels between the mechanisms of forward propagation and error backpropagation in deep neural networks and the reafferent (motor feedback) and efferent (motor command) signals found in neuroscience.

If an agent's actions were to be influenced by these two signals, the origin of actions should likely reside somewhere other than the output layer, allowing sufficient layers for forward propagation and error backpropagation to emulate these two signals in neuroscience. To explore this hypothesis, we propose a radical approach: postulating that the source of actions might lie at the input layer, rather than the output.

Consequently, we model the agent as a deep neural network, where state and actions are treated as input data and reward as the output. However, updating actions via backpropagation in this manner is prone to encountering numerical local minima, as the agent's actions are factually performing gradient descent on an error surface created by a single deep neural network. To address this issue, we incorporate MWM-SGD, proposed by Wang [21], into the reinforcement learning framework.

We treat an agent as a neural ensemble $\mathbb{W}$ where $\mathbb{W}$ comprises m homogeneous neural networks and $\mathbb{W} = \{W_1, W_2, \ldots, W_m\}$. We view the state s as an input vector, the actions $a_{:t}$ as a sequence of input vectors, and the rewards $r_{:t}$ as a sequence of output vectors.

For each interval, when the neural ensemble observes present state s, the neural ensemble initializes future actions $a_{:t}$ where $a_{:t} = \{a_0, a_1, ..., a_{t-1}\}$. Then the neural ensemble iteratively and randomly selects $W_i \sim \mathbb{W}$ and updates $a_{:t}$ by error back-propagation:

$$a_{:t} \leftarrow a_{:t} - \beta \frac{\partial}{\partial a_{:t}} E\Big(r^*_{:t}, f\big(W_i, (s, a_{:t})\big)\Big) \tag{1}$$

where β is the updating rate, $r^*_{:t}$ are the desired rewards in the incoming future, E is the error or loss function, $\leftarrow$ is error backprop, imitating efferent signals (motor command signals), and $f(\cdot)$ is the forward function outputting envisaged rewards, imitating reafferent signals (motor feedback signals).

Then, the neural ensemble executes future actions $a_{:t}$, imitating the final stage in Readiness Potentials (RPs), and observes actual future rewards $r_{:t}$, after which s, $a_{:t}$ and $r_{:t}$ are stored into long term experience replay buffer $\mathbb{D}$:

$$\mathbb{D} \leftarrow \mathbb{D} \cup \{(s, a_{:t}, r_{:t})\} \tag{2}$$

Upon sufficient $\mathbb{D}$, for each W in $\mathbb{W}$, the neural ensemble iteratively and randomly selects $s_j, a_{j:j+t}, r_{j:j+t} \sim \mathbb{D}$ and updates W by error backprop:

$$W \leftarrow W - \alpha \frac{\partial}{\partial W} E\Big(r_{j:j+t}, f\big(W, (s_j, a_{j:j+t})\big)\Big) \tag{3}$$

where α is the learning rate. In offline learning, statement 3 is performed after each episode[3], and updated $\mathbb{W}$ will be used in the next episode, forming a circulation.

[3] Or after a predetermined number of episodes, known as delayed learning.

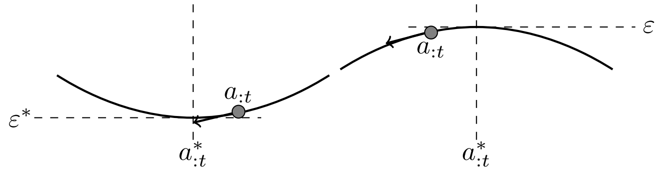

Figure 1: **Left**: Suppose the neural nets are initialized as $\inf_{s,a_{:t}} E\big(r^*_{:t}, f(s,a_{:t})\big) \geq \varepsilon^*$ where ε^* is positive, the future actions $a_{:t}$ of the agent will roll into the low point $a^*_{:t}$ where the agent prematurely thinks $a^*_{:t} = \operatorname{argmin}_{a_{:t}} E\big(r^*_{:t}, f(s,a_{:t})\big)$ and $f(s,a^*_{:t}) \approx r^*_{:t}$. **Right**: However, after learning $f(s,a^*_{:t}) = r_{:t}$ where $r_{:t} \neq r^*_{:t}$ and $E\big(r^*_{:t}, f(s,a^*_{:t})\big) = \varepsilon > \varepsilon^*$, under similar s, the future actions $a_{:t}$ of the agent will roll away from the high point $a^*_{:t}$, encouraging the agent to find or explore alternative path to desired rewards in a numerical way.

The above method is based on the following numerical reason: upon observing present state s, an agent's future actions $a_{:t}$ are performing stochastic gradient descent among error surfaces[4] created by neural ensemble $\mathbb{W}$ for $f(s,a_{:t})$ to approximate desired rewards $r^*_{:t}$, as shown in Figure 1 (left). However, when the environment later reveals the actual future rewards $r_{:t} \neq r^*_{:t}$, $\mathbb{W}$ will adjust itself by learning $s, a_{:t}, r_{:t}$ so that $f(s,a_{:t})$ will approximate actual future rewards $r_{:t}$ where $r_{:t} \neq r^*_{:t}$ and level up the previous supposed minimum. After learning, upon observing present state similar to s, the agent's $a_{:t}$ will be performing stochastic gradient descent among updated error surfaces created by updated neural ensemble $\mathbb{W}$ for $f(s,a_{:t})$ to approximate desired rewards $r^*_{:t}$. However, since the error surfaces are updated, the agent's gradient of $a_{:t}$ is also updated and will roll away form the previous supposed minimum as shown in Figure 1 (right).

Under this trial and error circulation, $\mathbb{W}$ are constantly changing, as well as the error surfaces upon which future actions $a_{:t}$ are updated. The learning process will continue until the supposed minima converge to the actual minima and actions $a_{:t}$ constantly reach the actual and supposed minima, leaving little error to propagate for the neural nets to learn.

Upon this view, the initialization of the agent's future actions $a_{:t}$ can be viewed as analogous to ϵ-greedy method for balancing exploration and exploitation. Specifically, when $a_{:t} \sim \mathcal{N}(0,\sigma^2)$ (i.i.d.) or $a_{:t} \sim \mathcal{U}(-\sigma,\sigma)$ (i.i.d.), the σ functions as the ϵ in ϵ-greedy approach. A larger σ induces greater variability in the agent's actions, even after fine-tuning, thereby facilitating exploration beyond environmental local minima. Conversely, a smaller σ reduces action variability, thereby encouraging exploitation around known environmental local minima.

Since the future actions of an agent are iteratively generated from input layer of deep neural networks rather than output, we refer to this model as Reversal

[4] Or non-convex high-dimensional error surfaces.

Generative Reinforcement Learning (RGRL).

We present a more sophisticated algorithm for offline learning in Algorithm 1:

Algorithm 1 RGRL with offline learning

```
Initialize homogenious neural ensemble $\mathbb{W}$ where $\mathbb{W} = \{W_1, W_2, \ldots, W_m\}$
Initialize long term experience replay buffer $\mathbb{D}$
for each episode do
    Initialize short term experience replay buffer $D$
    Initialize environment
    while not done do
        Observe present state $s$
        Initialize future actions $a_{:t} = \{a_0, a_1, ..., a_{t-1}\} \sim \mathcal{U}(-\sigma, \sigma)$ (i.i.d.)
        Initialize desired rewards $r^*_{:t}$
        for each iteration do
            Select $W_i \sim \mathbb{W}$
            Back-prop $a_{:t} \leftarrow a_{:t} - \beta \frac{\partial}{\partial a} E\Big(r^*_{:t}, f\big(W_i, (s, a_{:t})\big)\Big)$
        end for
        Observe and execute future action $a_0$
        Observe future reward $r_0$ resulting from $a_0$
        Store $s, a_0, r_0$ to $D$
    end while
    Return agent performance to human for inspection
    Sequentialize $D$ to $\mathbb{D}$
    Drop duplicated or similar experience in $\mathbb{D}$
    for $W$ in $\mathbb{W}$ do
        for each iteration do
            Select $s_j, a_{j:j+t}, r_{j:j+t} \sim \mathbb{D}$
            Back-prop $W \leftarrow W - \alpha \frac{\partial}{\partial W} E\Big(r_{j:j+t}, f\big(W, (s_j, a_{j:j+t})\big)\Big)$
        end for
    end for
    Use stochastic eviction policy to limit the size of $\mathbb{D}$
end for
```

where sequentializing D to $\mathbb{D}$ is defined as: in D, for s in time step j, $a = \{a_j, a_{j+1}, ..., a_{j+t-1}\}$, $r = \{r_j, r_{j+1}, ..., r_{j+t-1}\}$, and we store s, a, r to $\mathbb{D}$ with fixed t. Namely, we use fixed-size rolling window to subtract pairs of s, a, r from D into $\mathbb{D}$.

To further help the agent escape from environmental local minima, in addition to initializing with a large σ, we also adopt a periodic exponential annealing-and-reset strategy. Specifically, in the first few episodes, the agent starts with high σ, which is gradually reduced in an exponential decay fashion over a series of episodes. After a predefined cycle of episodes, the σ is reset to a high initial value again, and the decay process restarts. This cycling initialization of σ of decay and reset is designed to allow the agent to periodically regain high exploratory capacity, enabling it to escape from local minima. The cycle length used in our implementation is set to 10 episodes for ease of verification across all environments.

3 Experimental Results

To validate the above methodology, given that the process exclusively involves neural networks, a wide variety of architectures can be employed as substitutes for $f(s, a_{:t})$, provided that error propagation can be effectively traced back to the input layer. For the sake of simplicity and demonstration purpose, in this work, we evaluate the approach using traditional RNN by Rumelhart et al. [16], GRU by Cho et al. [3], LSTM by Hochreiter et al. [9] and Transformer Decoder by Vaswani et al. [20] independently. Also, to test the robustness of the proposed approach, we initialize each episode with random seed.

Since RL problems typically involve long-term dependencies where the agent needs to understand and predict how action taken in the present state will affect future rewards and future states and how future action taken in the future state will affect thereafter. Therefore, the future states should be better envisaged ahead by the agent to better imagine or envisage future rewards. To achieve this goal, we modified the above auto-regressive models in a feedback manner where the inputs are state-action pairs and the outputs are reward-state pairs in each time step. The state in each input are the direct forwarded state resulted from the output from the previous step (feeding the output back to the next input). These feedback autoregressive models initially take the first pair of state-action and then generate the first reward-state pair. Then these models will take the first and second pair of state-action to generate the second pair of reward-state (so on and so forth). To improve its computational efficiency, we incorporated key-value (KV) cache, commonly used in Transformer auto-regressive models by caching the attention keys and values of previous steps and allowing only the present step to be taken into computation. This method helps the agent to imagine or envisage future states where future states might not be visible from the present state and also helps the agent to form a logic chain to capture the causal dynamics of the environment[5].

We present the experimental results concerning the the impact of the neural ensemble—specifically, the size of m in $\mathbb{W}$ where $\mathbb{W} = \{W_1, W_2, \ldots, W_m\}$—on RNN, GRU, LSTM and Transformer Decoder (TD) with feedback.

Figure 2 shows the impact of the size of m. By increasing the size of m, namely the size of the neural ensemble, the agent reaches peak reward more quickly.

We observe that the use of multiple neural networks can minimize the numerical local minima created by each single neural net, stabilizing the gradient descent process of the agent's actions, and helping the agent to quickly infer and learn around critical regions.

[5] For future states to be taken into consideration, line 24 in Algorithm 1 becomes $W \leftarrow W - \alpha \frac{\partial}{\partial W} E\Big(\big(r_{j:j+t}, s_{j+1:j+1+t}\big), f\big(W, (s_j, a_{j:j+t})\big)\Big)$. And since future states are not visible to the agent during planning in line 12 and are instead envisaged by the agent, line 12 in Algorithm 1 will remain the same.

Since solving the issue of numerical local minima can lead to the agent becoming more inclined toward exploitation, a larger m should be coupled with larger initial σ to balance exploitation and exploration and to minimize likelihood of the agent getting stuck at environmental local minima.

We demonstrate that the process of assigning or predicting future rewards based on the present state and future actions can be modeled by deep neural networks, thus mitigating the reliance on the Bellman equation or other functions designed by experts.

4 Future Works

In this work, we introduce a basic framework work purely for deep neural networks to engage in reinforcement learning. However, in the present introductory work, we present only offline learning. Furthermore, due to limitation in hardware resource, we take vectorized state as input rather than raw image. Also, the neural ensemble shares identical desired rewards $r^*_{:t}$ and comprises homogeneous neural net, which might not entirely true for human. Finally, we postulate that, by deploying multiple agents in the same digital or physical environment concurrently[6], the data gathering process can be accelerated.

5 Conclusion

In this paper, we present a reinforcement learning model that is both model-free and value-function-free, replacing the traditional Bellman equation purely with deep neural networks. We try to find an alternative approach to mainstream perspectives: first, we demonstrate that deep neural network is capable of deriving information from both their output and input layers. Second, we demonstrate that actions of an agent can be derived from input layer of neural networks using iterative forward and backward propagation. Third, we show that by iteratively training the networks using self-generated data from its own input and its interaction with environment, the networks can dynamically adapt to changing environmental conditions. Fourth, we demonstrate that the proposed method can be seamlessly integrated with other state-of-the-art architectures or methodologies.

Acknowledgments

We thank reviewers for their insightful discussions and feedback.

[6] This approach is a little different from federated learning by McMahan et al. [12] in that it is the global models doing training and local models doing data gathering. However, we don't rule out the possibility of incorporating original federated learning such as the global models of this paper becomes the local models of McMahan et al. [12], enabling cross border data gathering and training.

Conflict of interest statement

No funding was received to assist with the preparation of this manuscript.

References

[1] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z Leibo, Jack Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *arXiv:1606.04460v1*, 2016.

[2] Yize Chen, Yuanyuan Shi, and Baosen Zhang. Modeling and optimization of complex building energy systems with deep neural networks. *51st Asilomar Conference on Signals, Systems, and Computers*, 2017.

[3] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[4] L. Deecke. Planning, preparation, execution, and imagery of volitional action. *Cognitive Brain Research, 3(2), 59-64*, 1996.

[5] Michel Desmurget and Scott Grafton. Forward modeling allows feedback control for fast reaching movements. *Trends in Cognitive Sciences – Vol. 4, No. 11*, 2000.

[6] P Haggard and M Eimer. On the relation between brain potentials and the awareness of voluntary movements. *Experimental Brain Research, 126(1), 128-133*, 1999.

[7] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. *arXiv:1709.06560v3*, 2019.

[8] G.E. Hinton. How neural networks learn from experience. *Scientific American*, pages 145–151, 1992.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation, Volume 9, Issue 8*, 1997.

[10] H.H. Kornhuber and L. Deecke. Brain potential changes in voluntary and passive movementsin humans: readiness potential and reafferent potentials. *Pflügers Archiv'S Historical Articles*, 2016.

[11] B Libet, C A Gleason, E W Wright, and D K Pearl. Time of conscious intention to act in relation to onset of cerebral activity (readiness-potential). the unconscious initiation of a freely voluntary act. *Brain, 106(3), 623-642*, 1983.

[12] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop*, 2013.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, and Georg Ostrovski. Human-level control through deep reinforcement learning. *Nature, 518(7540):529–533*, 2015.

[15] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going deeper into neural networks. In *Archived from the original on 2015-07-03*. Google Research, 2015.

[16] David E. Rumelhart, Geoffrey E. Hinton, and James L. McClelland. Learning internal representations by error propagation. In *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*. MIT Press, 1987.

[17] Aaron Schurger, Jacobo D Sitt, and Stanislas Dehaene. An accumulator model for spontaneous neural activity prior to self-initiated movement. *Proceedings of the National Academy of Sciences, 109(42), E2904-E2913*, 2012.

[18] Miguel Suau, Alexandros Agapitos, David Lynch, Derek Farrell, Mingqi Zhou, and Aleksandar Milenovic. Offline contextual bandits for wireless network optimization. *arXiv:2111.08587v1*, 2021.

[19] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Conference on Neural Information Processing Systems (NIPS)*, 2000.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Conference on Neural Information Processing Systems (NIPS)*, 2017.

[21] Brown Wang. Deducing decision by error propagation. *Proceedings of Proc. of the Adaptive and Learning Agents Workshop (ALA)*, 2022.

[22] D.M. Wolpert and M. Kawato. Multiple paired forward and inverse models for motor control. *Neural Networks Volume 11, Issues 7–8*, 1998.

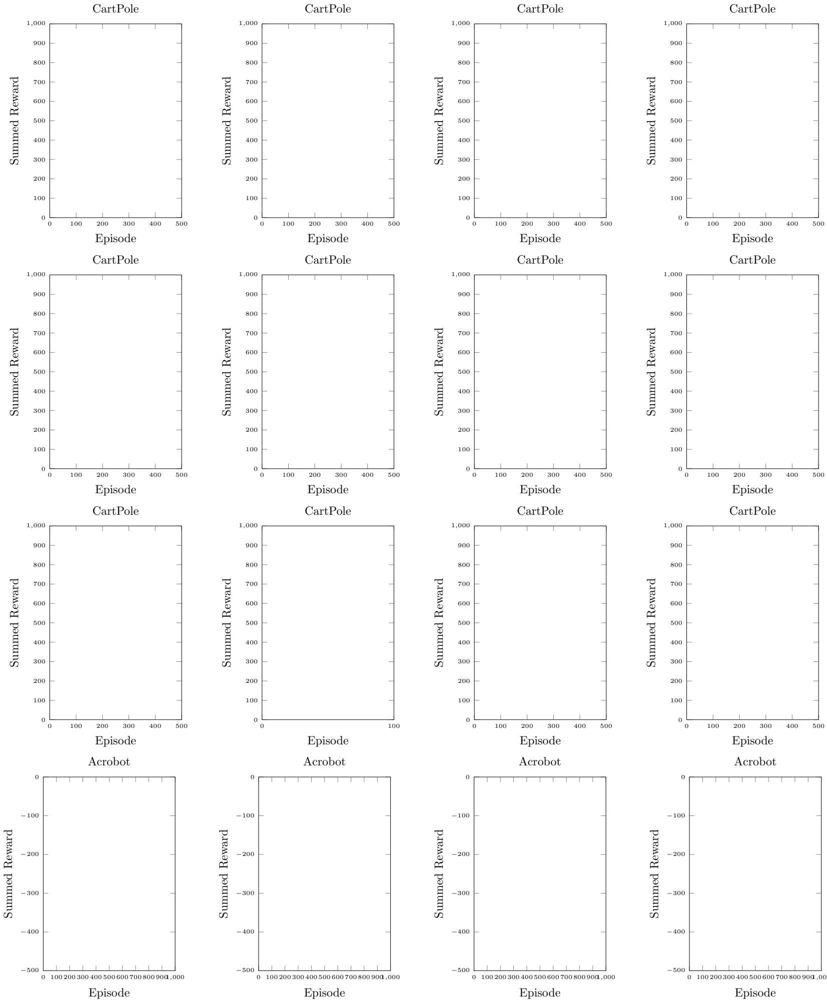

Figure 2: Performance comparison under different neural types and size of m with random seed for each episode