Deep Reinforcement Learning Experiments in the Collector Environment

Emrecan Ilik
Industrial Engineering and Management(M.Sc.)
Informationsdienste und elektronische Märkte
Karlsruher Institut für Technologie (KIT)
`uxyes@student.kit.edu`

Abstract

Deep Reinforcement Learning has been gaining importance in the field of Artificial Intelligent. Grid world environments enable researchers to conduct Reinforcement Learning experiments and test the capabilities of Reinforcement Learning algorithms. The collector environment is a grid-based environment wherein an agent performs tasks while moving through a grid world. On top of that, the collector environment provides an opportunity to test and benchmark Deep Reinforcement Learning algorithms. Deep Q-Network (DQN), and Proximal Policy Optimization (PPO) are two popular Deep Reinforcement Learning algorithms in the literature. Thus, this study engages in performing Deep Reinforcement Learning experiments by employing DQN and PPO. The experiments focus on implementing DQN and PPO, moreover benchmarking their performance, and modifying the training envionment to monitor and interpret how the performance of the agents changes.

1 Introduction

"Reinforcement Learning (RL) deals with solving sequential decision-making problems. Many real-world problems such as playing video games, driving, and robotic control can fall into that category. These problems can be solved by humans and machines" (Graesser and Keng, 2019). Furthermore, the study by Arulkumaran et al. points out that Deep Reinforcement Learning (DRL) is inclined to transform the field of Artificial Intelligence (AI) and, it demonstrates a groundbreaking improvement that can give way to the implementation of more intelligent systems.

Moreover, the Minigrid library which was developed by Farama Foundation (2021) provides grid-world environments to carry out research on reinforcement learning. The collector environment by Schweizer (2022) was developed on the Minigrid wherein an agent collects items while moving inside a grid world. The environment proposes an opportunity to test the capabilities of RL algorithms. To give an example, the research by Mnih et al. (2015) illustrates the implementation of the Double Deep Q-Learning (DQN) algorithm in numerous games. On top of that the research by Mnih et al. highlights that DQN can outshine many RL algorithms mentioned in the literature. Moreover, the study by Schulman et al. indicates that the Proximal Policy Optimization (PPO) algorithm shows potency in different environments.

This paper aims to test the effectiveness of DQN and PPO algorithms in the frame of the collector environment. Moving ahead of the conclusions of the existing research papers, this work attempts to investigate how DQN and PPO algorithms perform in the collector environment. The capacity and shortcomings of these algorithms in the framework of the collector environment, and if the performance of the algorithms can be enhanced through modifications are the focus points. To examine these points some experimental tests are to be conducted within the scope of this work. Beyond that, this paper touches upon the impacts of reward scaling and, the effect of the episode length in the collector environment.

The conclusions of this work can shed light on the capabilities of DQN and PP0 algorithms in grid-based environments. In brief, this study aims to analyze and compare the performance of DQN and PPO algorithms in the collector environment.

2 Deep Reinforcement Learning

The purpose of this chapter is to demonstrate an overview of Deep Q-Network (DQN) and Proximal Policy Optimization (PPO) algorithms and stress their relevance for the study.

Reinfrocement Learning (RL) is about learning which actions to perform and how to transform situations into actions to maximize a numerical reward signal. An RL agent is not told which actions to perform, rather it learns which actions output the most reward by trying them (Sutton and Barto, 2018). Beyond that, current improvements in deep learning have made it achievable to extract high-level features from raw data which gave way to a step forward in computer vision (Mnih, 2013; Sermanet et al., 2013; Krizhevsky et al. 2017). Additionally, conclusions by Mnih et al. point out that those methods are based on the utilization of some neural network architectures which mainly are convolutional neural networks, multilayer perceptrons, and recurrent neural networks, whereby one should ask if the same technic can be beneficial for RL. Improvements in computer vision have come into view due to efficient training of deep neural networks on large data sets (Krizhevsky et al., 2017).

The research Mnih et al. (2013) states that deep learning and its success encouraged Mnih et al. to combine RL with a deep neural network that can process training data successfully. Consequently, the study by Mnih et al. proposes the “Deep Q Learning” (DQN) algorithm. Beyond that, the study by Mnih et al. compares DQN with the best RL algorithms in the literature. According to the finding of the study, DQN outperforms many RL algorithms in various games.

On the other hand the study by Schulman et al. mentions that Q-learning can perform poorly on many simple tasks and vanilla policy gradient methods are not data-efficient, and they demonstrate poor robustness. Moreover, trust region policy optimization (TRPO) is complicated, and it does not match up with architectures that contain noise (Schulman et al., 2017). To tackle these problems, the research by Schulman et al. (2017) aims to introduce an algorithm that is data-efficient and yields reliable performance of TRPO by using first-order optimization. Thus, they propose the “Proximal Policy Optimization Algorithms”(PPO) (Schulman et al., 2017). Their research also compares PPO to other algorithms from the literature.

To summarize, one can conclude the following arguments from the studies Mnih et al. (2013), Mnih et al. (2015), and Schulman et al. (2017): DQN showcased its efficiency in various Atari games. It can learn optimal Q values, enabling it to make the right decisions in numerous environments. In addition to that, it has the ability to handle discrete action spaces. Besides that PPO represents effectiveness in handling continuous action spaces and stochastic policies. The Trust Region Policy Optimization method which PPO employs enables the algorithm to make better decisions.

Through the employment of DQN and PPO, this paper aims to monitor and benchmark their performance in the collector environment. The significance and relevance of these algorithms in RL experiments have been discussed in this chapter. Experimental results which are to be displayed in the following chapter will provide further insights on the abilities and performance of these algoritms. The next chapter sheds light on the relevance of Stable-Baselines3 and Sacred for the RL experiments.

3 Experimental Setup

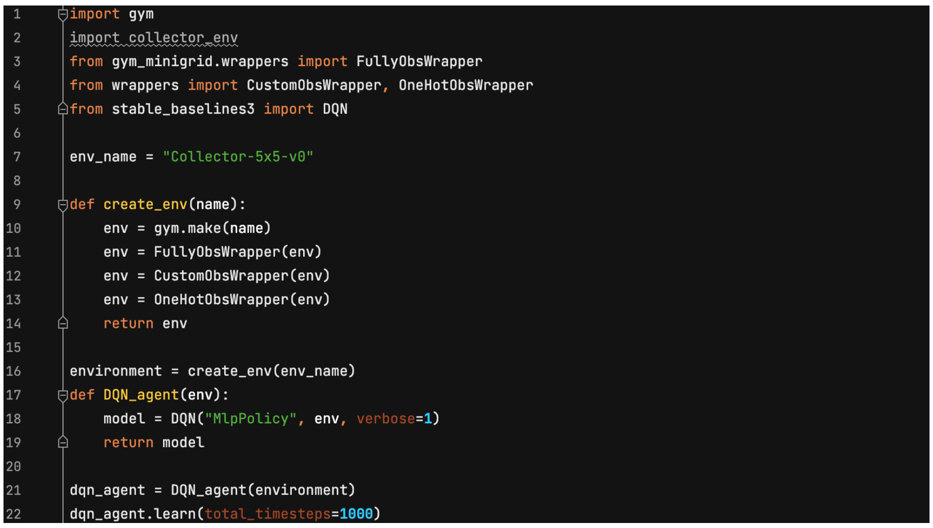
This chapter deals with important facades of experimental setup by using Stable-Baselines3 with DQN and PPO in the collector environment. Stable-Baselines3 enables the implementation of many RL algorithms including DQN and PPO. On the other hand, the chapter also highlights other important components of the experimental setup, namely reproducibility and the environment design.

First, the chapter provides insights into the properties of Stable-Baselines and demonstrates how the implementation of RL algorithms can be realized by using the Stable-Baselines3 framework. Next, the chapter focuses on the reproducibility of RL experiments. It explains the importance of the reproducibility of machine learning experiments, and implementation thereof by using Sacred. Finally, the chapter demonstrates the collector environment and its properties.

3.1 Stable-Baselines3: A Useful Tool to Conduct RL Experiments

To begin with, analysis by Raffin et al. states that even though deep reinforcement learning research has made groundbreaking improvements in recent years, the reproducibility of experimental results still poses a challenge. The authors indicate that the reliability of experimental baselines is vital, otherwise novel algorithms compared to weak baselines can yield unreasonably high estimates of performance improvements (Raffin et al., 2021a).

To cope with this problem, the research by Raffin et al. (2021a) suggests Stable-Baselines3 (SB3), an open-source framework for implementing deep reinforcement learning algorithms. By proposing Stable-Baselines3, it is persuaded by Raffin et al. (2021a) the aim of providing a reliable and user-friendly RL library(Raffin et al., 2021b). An implementation of DQN with Stable-Baselines3 interfaced with the collector environment is represented in the figure below.



```python
import gym
import collector_env
from gym_minigrid.wrappers import FullyObsWrapper
from wrappers import CustomObsWrapper, OneHotObsWrapper
from stable_baselines3 import DQN

env_name = "Collector-5x5-v0"

def create_env(name):
    env = gym.make(name)
    env = FullyObsWrapper(env)
    env = CustomObsWrapper(env)
    env = OneHotObsWrapper(env)
    return env

environment = create_env(env_name)
def DQN_agent(env):
    model = DQN("MlpPolicy", env, verbose=1)
    return model

dqn_agent = DQN_agent(environment)
dqn_agent.learn(total_timesteps=1000)
```

Figure 1: Implementation of DQN on top of Stable-Baselines3

The article by Raffin et al. mentions that the researchers provide a reliable, clean, and fully documented API, inspired by the scikit-learn API (Pedregosa et al.,2011). According to Raffin et al. features of SB3 can be listed as :

- Simple API
- Documentation
- High-Quality Implementations
- Comprehensive
- Experimental Framework

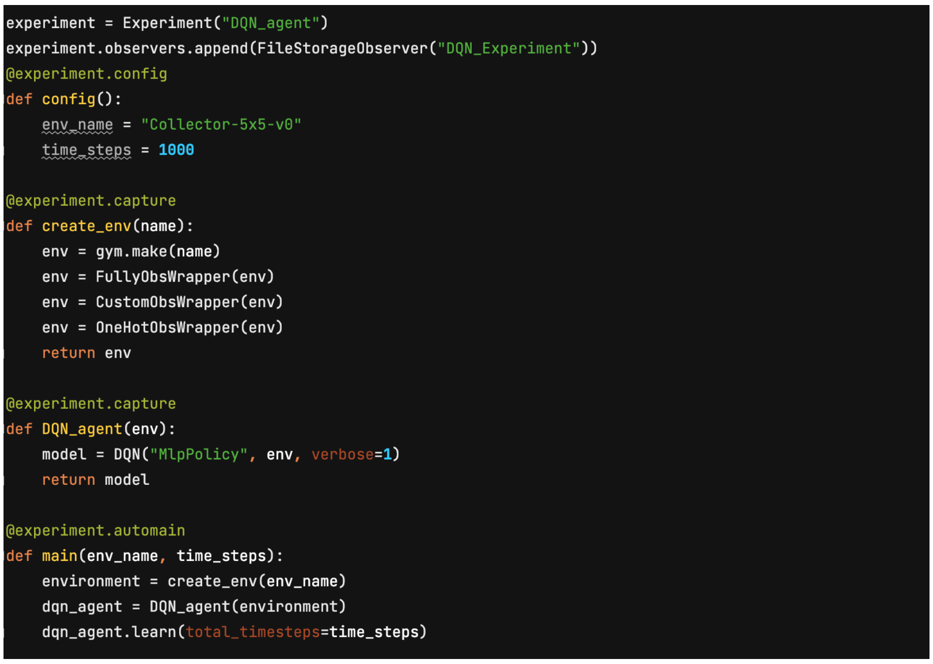
To conclude, Stable-Baselines3 provides a tool that enables easy and robust implementation of RL algorithms to researchers. For those reasons, Stable-Baselines3 is also preferred to conduct RL experiments in this work. Having highlighted the implementation of RL algorithms with Stable-Baselines3, another aspect of RL experiments namely, reproducibility should be explained. Thus, the next subsection provides insights into the concept of reproducibility.

3.2 Reproducibility in RL Experiments

According to Greff and Schmidhuber (2015) machine learning (ML) research encompasses lots of manual work in addition to actual innovation. This may include making implementations configurable, hyperparameter tuning, saving experimental parameters and results, and safeguarding that they are reproducible (Greff and Schmidhuber, 2015). On top of that, one can in accordance with the observation by Henderson et al. say that it is rarely easy to reproduce the results of deep RL experiments. Thus, reproducibility can be undervalued under the pressure of delivering an ML project on time, although it is one of the important elements of any ML project. (Greff and Schmidhuber, 2015).

As a consequence of the existing gap in addressing the reproducibility issue without exerting a high level of effort, the research by Greff and Schmidhuber proposes Sacred as an open-source Python library. They mention that the main goal of Sacred is to provide help for the problems mentioned above, which requires a minimal adjustment of code (Greff and Schmidhuber, 2015).

To sum up, the findings of Greff and Schmidhuber (2015) are indicating that Sacred mainly aims to collect all the needed information in order to make all runs of experiments reproducible. This work therein uses Sacred to ensure the reproducibility of RL experiments. The figure below illustrates an implementation of Sacred for DQN.



```
experiment = Experiment("DQN_agent")
experiment.observers.append(FileStorageObserver("DQN_Experiment"))
@experiment.config
def config():
    env_name = "Collector-5x5-v0"
    time_steps = 1000

@experiment.capture
def create_env(name):
    env = gym.make(name)
    env = FullyObsWrapper(env)
    env = CustomObsWrapper(env)
    env = OneHotObsWrapper(env)
    return env

@experiment.capture
def DQN_agent(env):
    model = DQN("MlpPolicy", env, verbose=1)
    return model

@experiment.automain
def main(env_name, time_steps):
    environment = create_env(env_name)
    dqn_agent = DQN_agent(environment)
    dqn_agent.learn(total_timesteps=time_steps)
```

Figure 2: An Implementation of Sacred

Having touched upon the reproducibility aspect of RL experiments, the characteristics of the training environment are to be explained in the next subsection.

3.3 Collector Environment

The collector environment was developed by Schweizer (2022) through using the Minigird library. Properties of the environment are explained by Schweizer (2022) as follows: ”In the collector environment, a single agent lives in a grid world that is in possession of two items, namely a blue ball and a green key. The agent is tasked to pick those items up. One instance of each item exists in the environment at each point in time.

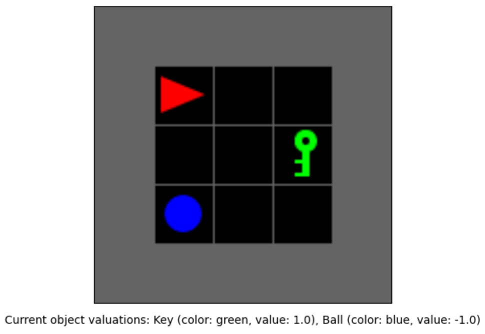

Current object valuations: Key (color: green, value: 1.0), Ball (color: blue, value: -1.0)

Figure 3: collector-env Schweizer (2022)

In case an item is collected, the same sort of item comes into view again at a random, distinct location. If an item is not collected, the position of the item remains the same. Furthermore, the environment is fully-observable. The environment is deterministic, except for the item placement. Moreover, the agent can perform 4 actions, namely “turn left”, “move forward”, and “pick up an object”. The figure below illustrates the action space of the environment.

On top of that item values can take on two integers which are -1 and 1. When one type of item has a value of 1, the other one has -1 or vice versa. Beyond that valuation of the items change over time i.e. from 1 to -1 or from -1 to 1. In addition to that, an episode ends after a given number of steps which is parametrized by max-steps”Schweizer (2022).

The algorithms employed in RL experiments, the tools, and the training environment have been explained so far. The next chapter focuses on experimental questions, hypotheses, and experimental results.

4 Experimental Results

This chapter focuses on performing experiments with DQN and PPO in the collector environment. The performance of these algorithms is to be evaluated and analyzed. The experiments to be conducted concern the magnitude of the reward signal received by the agent and the impact of episode length on the overall performance of the RL agents. For instance, the research by van Hasselt et al. (2016) mentions the role of the magnitude of the reward in RL. In addition to that the study by Pardo et al. (2018) shows empirically that the time factor significantly affects the performance of PPO and it can cause situations wherein the algorithm can represent interesting decisions.

All in all conducting these experiments is important to gain insight into the behaviors of DQN and PPO. The results of these experiments provide conclusions on the impact of reward scale and episode length on the performance of RL algorithms in grid environments.

Question: How would DQN and PPO behave in the collector environment? Could one of the algorithms yield better performance than the other?

Hypothesis: The study by Schulman et al. (2017) highlights drawbacks of DQN. It points out that PPO can handle both discrete and continuous action spaces. Considering the dynamic components of the training environment, it is hypothesized that PP0 will yield better performance than DQN in terms of average reward per episode.

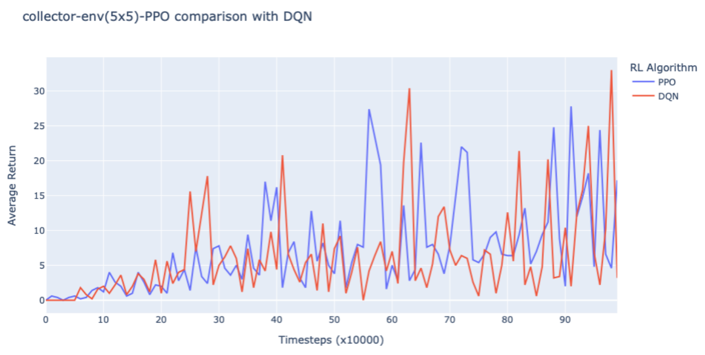


Figure 4: PPO and DQN comparison during training

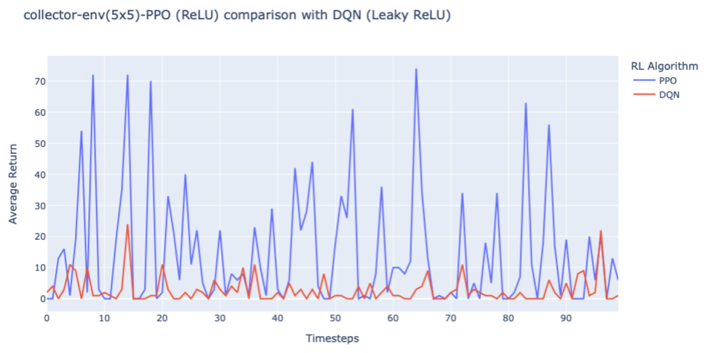


Figure 5: PPO and DQN comparison after training

Both agents trained for 1 million time steps. Then they were evaluated during and after training. During training, PPO reached an average reward per episode of 7.36 while DQN's overall score is 6.21. However, PPO represents a much higher performance than DQN after training.

Question: How does changing the reward scale from 1,-1 to 10,-10, and reducing the episode length from 200 to 100 impact the performance of the agent?

Hypothesis: Changing the magnitude of the reward from 1,-1 to 10,-10 incentivizes the agents. Furthermore, reducing the episode length applies time pressure. It is hypothesized that the performance of DQN and PPO will significantly increase, and PPO will demonstrate a better performance than DQN.

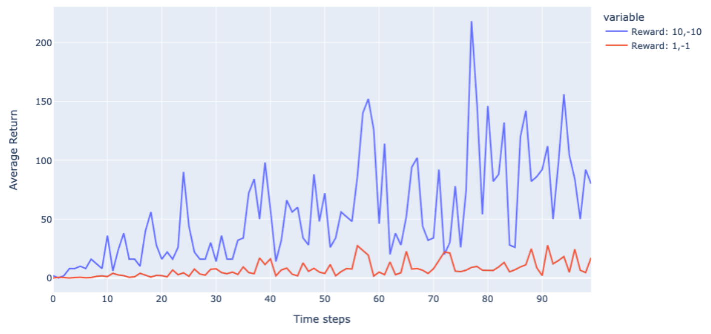


Figure 6: PPO - Evaluation during training

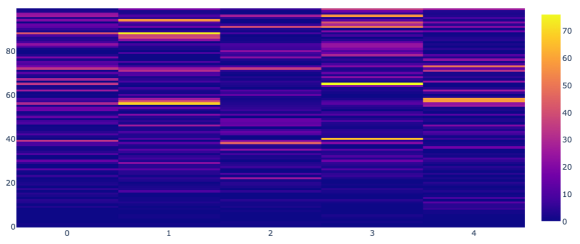


Figure 7: PPO - Heapmat of Reward (-1,1)

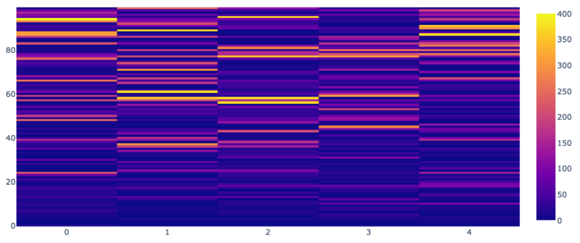


Figure 8: PPO - Heapmat of Reward (-10,10)

Experimental results show that changing the magnitude of the reward from -1,1 to -10,10 enabled PP0 to demonstrate better performance. The results highlight the importance of reward scala and episode length on the overall performance of PPO. Furthermore, if we compare the heatmap of reward with -1,1 and -10,10, we can conclude that with the reward scale of -10,10, PPO reached much higher reward values.

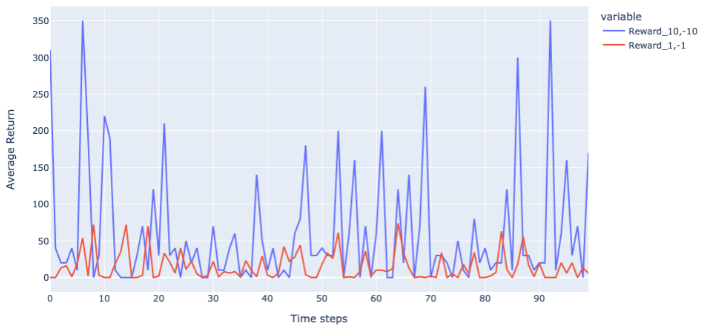


Figure 9: PPO - Evaluation after training

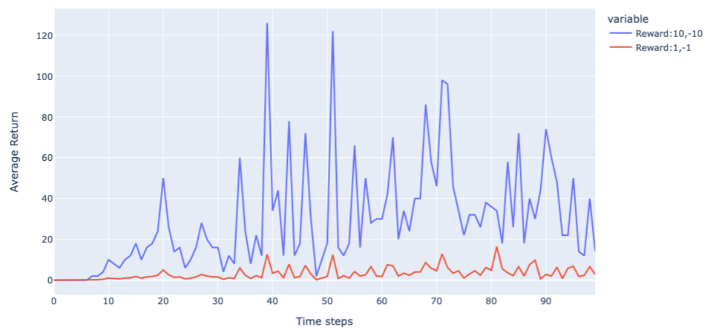


Figure 10: DQN - Evaluation during training

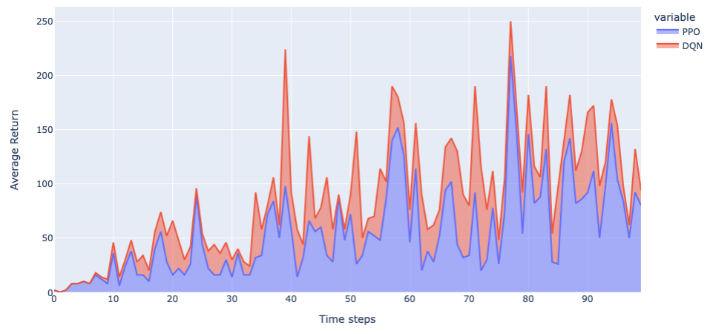


Figure 11: PPO comparison to DQN - Evaluation during training

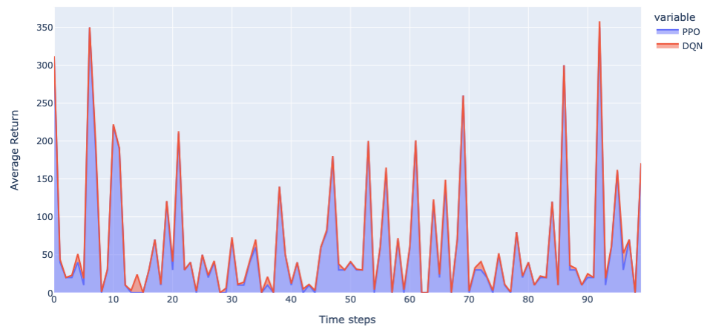


Figure 12: PPO comparison to DQN - Evaluation after training

Experimental results indicate that rescaling the reward and reducing the episode length have a remarkable impact on the agents. Besides that, the overall performance of both of the agents increased. On top of that, PPO demonstrated better performance than DQN both during and after training. Especially, after training PPO outperformed DQN. The next chapter highlights the evaluation of the experimental results and provides an overview of the conclusions of the study.

5 Conclusion

To sum up, the performance of PPO and DQN in a grid world environment, namely the collector environment was analyzed in this study. By performing experiments, some vital conclusions have come into view. The first conclusion is that both DQN and PPO are capable of performing tasks while they travel through a grid-based environment. Both algorithms increased their accumulated reward over time. As a second conclusion it is worth mentioning that PPO demonstrated better performance than DQN. Furthermore, it emerges that reward scale and episode length make a significant impact on the performance of DQN and PPO. For instance, PPO returned an average accumulated reward of 55.64 which increased from 7.36 after the reward scale had been modified from -1,1 to -10,10. Similarly, the average accumulated reward of DQN improved from 3.23 to 29.68 after the modification of the reward and episode length. This shows that incentivizing RL agents through implementing higher rewards can enable the agents to yield better performance.

Besides that, we can state that the quality of training is contingent upon several factors. For example, total time steps, and the hyperparameters affect the training process. All in all, the conclusions from the experiments provide a better understanding of the behavior and the capabilities of DQN and PPO in grid world environments.

Eigenständigkeitserklärung

Ich versichere wahrheitsgemäß, die Arbeit selbstständig angefertigt, alle benutzten Hilfsmittel vollständig und genau angegeben und alles kenntlich gemacht zu haben, was aus Arbeiten anderer unverändert oder mit Abänderungen entnommen wurde.

Karlsruhe, den July 9, 2023

Emrecan Ilik

References

Arulkumaran, K., Deisenroth, M.P., Brundage, M., and Bharath, A.A. (2017) Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6), pp. 26–38.

Farama Foundation (2021) Minigrid: An openai gym environment for gridworld reinforcement learning. [GitHub repository], `https://github.com/Farama-Foundation/Minigrid`.

Graesser, L. and Keng, W.L. (2019) *Foundations of deep reinforcement learning.* Addison-Wesley Professional.

Greff, K. and Schmidhuber, J. (2015) Introducing sacred: A tool to facilitate reproducible research. *Proceedings of the AutoML. International Machine Learning Society.*

van Hasselt, H.P., Guez, A., Hessel, M., Mnih, V., and Silver, D. (2016) Learning values across many orders of magnitude. *Advances in neural information processing systems*, 29.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018) Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

Krizhevsky, A., Sutskever, I., and Hinton, G.E. (2017) Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), pp. 84–90.

Mnih, V. (2013) *Machine learning for aerial image labeling.* University of Toronto (Canada).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013) Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602.*

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

Pardo, F., Tavakoli, A., Levdik, V., and Kormushev, P. (2018) Time limits in reinforcement learning. In *International Conference on Machine Learning*, pp. 4045–4054, PMLR.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011) Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12, pp. 2825–2830.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021a) Stable-baselines3: Reliable reinforcement learning implementations. *The Journal of Machine Learning Research*, 22(1), pp. 12348–12355.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. (2021b) Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268), pp. 1–8, URL `http://jmlr.org/papers/v22/20-1364.html`.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347.*

Schweizer, M. (2022) Collector-env: Gym environment for data collection. `https://github.com/mschweizer/collector-env`, [GitHub repository].

Sermanet, P., Kavukcuoglu, K., Chintala, S., and LeCun, Y. (2013) Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3626–3633.

Sutton, R.S. and Barto, A.G. (2018) *Reinforcement learning: An introduction.* MIT press.