Deep Q-network Development and Validation based on Interactive 2D Video Games

Cunhan You
ShanghaiTech University
youch2022@shanghaitech.edu.cn
2022233290

Zixian Zhang
ShanghaiTech University
zhangzx12022@shanghaitech.edu.cn
2022233316

Yi Wang
ShanghaiTech University
wangyi2022@shanghaitech.edu.cn
2022233255

Ruiqian Li
ShanghaiTech University
lirq2022@shanghaitech.edu.cn
2022133196

Abstract

Deep Q-Network (DQN), a type of reinforced deep learning, approximates a state-value function in a Q-Learning framework with a neural network, which is featured with lightweight dataset and none policy-based. This work shows an AI solution for Metroidvania games, taking advantage of the critic attributes of DQN. Deep neural network is used for approximating value-based function, meanwhile Q-learning save the value by data-table. We experimentally demonstrate an AI for Metroidvania games, which has strong aggressiveness and is more flexible and proactive in releasing skills in Hollow Knight. Particularly, since this model is based on the benefits of boss warfare, It can be easily ported to other Metroidvania games.

1. Introduction

Interactive 2D video games are currently a very popular type of game, characterized by building a 2D scene through a computer, where players control some characters or objects to operate and move. At the same time, the computer will advance the scene based on the player's behavior to generate interaction with the player, and finally ending with the player achieving certain conditions. The Metroidvania game is one of the most popular 2D video game, characterized by rich role-playing elements and focusing on action exploration. The most representative among them is the various of boss fights. The difficulty of boss fights in these game is very high, and ordinary players need to practice multiple times to pass. While at the same time, the process of boss fights is very enjoyable. Therefore, we hope to design an AI to complete this process.

After this section, we will introduce the principles of common game AI models and why DQN is chosen as the basic structure of our AI model. This structure will be shown in the third part, and introduced from three aspects: deep neural network, Q-learning algorithm and operation selection algorithm. Subsequently, in the fourth part, we will demonstrate the performance of the AI model in the Hollow Knight, a Metroidvania game, and limit the input and output data to verify its robustness. The fifth part mainly demonstrates the effect of migrating the model to other types of Metroidvania games, highlighting the difficulties and issues of this process. Finally, we will introduce the experiments and research directions that we will supplement in the future, as well as our workload distribution in the project process.

2. Related Work

2.1. Game AI

Currently, there are various types of AI for games available in the market, which can be broadly categorized into two main types. The first type utilizes traditional state-of-the-art (SOTA) algorithms, which are primarily based on the existence of unique solutions within the game itself. These AI systems employ simple algorithms that can compute the optimal next move for different game states. Classic examples of this type include the Tower of Hanoi and Tetris.

For more complex games, deep learning algorithms are commonly employed. There are two main approaches within this category. The first approach is based on convolutional neural networks (CNNs) driven by large datasets. This method involves training the model using a significant amount of pre-existing data, which is then directly used during actual gameplay, such as Mahjong [3].

The second approach is based on reinforcement learning with neural networks. This method involves setting up a reward mechanism to train the AI during gameplay. It is typically used in scenarios where the game environment is complex and there is limited availability of pre-training data. The game that we aim to develop aligns well with these characteristics. Therefore, we have chosen to reference the DQN method to train our AI.

2.2. Reinforcement Learning

Reinforcement learning, in contrast to CNN method, reduces the need for human knowledge and dataset. Instead, reinforcement learning does not rely on labeled data or pre-defined categories, and its objective is to maximize cumulative rewards through trial-and-error processes. Subsequently, the AI adjusts its network to increase the likelihood of successful runs.

The majority of deep reinforcement learning studies have been assessed within the Arcade Learning Environment (ALE), which offers emulated versions of numerous Atari games [1], such as Pong, Breakout, Space Invaders and so on. Additionally, ALE provides reward functions necessary for deep learning. Numerous algorithms have been introduced in the field of deep reinforcement learning. For instance, DQN were specifically designed to excel in the ALE, achieving performance comparable to humans [4]. DQNs function as regressions that learn to associate observations with expected rewards for each available action. The controller operates the DQN and selects actions with the highest expected value.

Based on the DQN approach for Atari games, students from Stanford University conducted development and validation of the Mario Kart game [2] in the course CS231, Deep Learning for Computer Vision. The results were highly impressive, as the DQN agent model achieved nearly the same performance as expert human players on typical race tracks. This success has greatly motivated us and increased our confidence in applying the DQN method to similar interactive 2D video games, especially Metroidvania games, with the aim of further showcasing its effectiveness.

3. Method

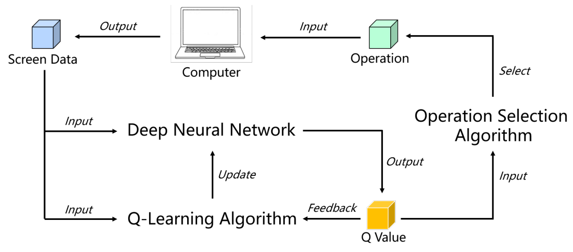

Figure 1. The Graph of Deep-Q-Network Structure for Metroidvania Games

Fig. 1 is the structural diagram of our DQN. As shown in the figure, we first capture image data from the computer screen and transmit them to the neural network and Q-learning algorithm. The former calculates a Q value. On the one hand, it will be fed back to the Q-learning algorithm and updated to the neural network. It will also be used as the input of the operation selection algorithm, so that the latter can output the best operation and return to the computer for the update of instructions.

3.1. Deep Neural Network in DQN

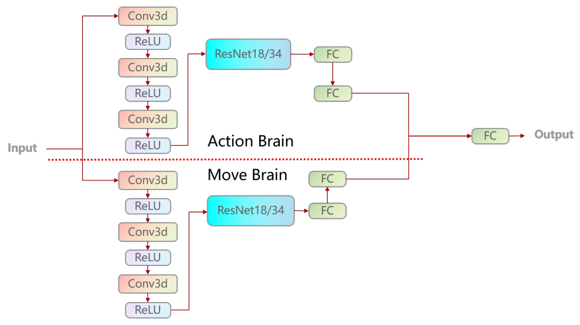

Figure 2. The Graph of Deep Neural Network in DQN

Fig. 2 is the structure of our neural network, which is divided into two parts. The upper part handles instructions for action, such as attack and dash, while the lower part handles instructions for movement, such as left and right. The network structure of these two parts is consistent, and processing them separately can separate these instructions without mix them together. Taking the upper part as an example: Firstly, we extract information from the image through a series of convolutional networks, and then obtain specific features through a ResNet, which are mapped onto operation instructions through two fully connected layers.

3.2. Q-Learning Algorithm in DQN

$$Q(s_{t+1}, a_{t+1}) = Q(s_t, a_t) + \alpha(r + \gamma max Q(s', a') - Q(s_t, a_t)) \quad (1)$$

Eq. 1 is our Q-learning algorithm. Among them, s_t represents the scene state at time t, at represents the operation instruction at time t, and Q is a special quantity given s and a, known as the Q value. α represents the learning rate, r represents the reward generated by the behavior, Q (s', a') represents the ideal Q value in the future, and γ then is the weight of future Q values. The reward part of traditional game AI's DQN usually only has time and the health of the character and boss, with the aim of training AI to defeat the boss as quickly as possible without injury. In order to accelerate the training process, we have added some additional rewards, such as the distance from the boss will determined whether skills will be used or not, which can help AI avoid battles less. In addition, we found in our experiments that in such games, current returns are more important than future expectations, so we reduced the value of discount and made AI pay more attention to the benefits of current operations.

3.3. Operation Selection in DQN

Algorithm 1 Epsilon-Greedy Algorithm

```
1: if random()< $\epsilon$ then
2:     Choose a random action a
3: else
4:     Choose the best action: $a = \arg\max_a Q(a)$
```

We adopted ϵ-greedy policy for operation selection. This logic is very simple, that is, if the random variable is less than ϵ. If so, random operations will be selected. In other cases, the corresponding instructions are selected based on the Q value, as Alg. 1. This can effectively prevent the training from falling into the local optimal solution. But at the same time, the efficiency of completely random exploration is very low, and it is very difficult to rely on completely random to perform good operations for network learning. Therefore, we have defined some basic knowledge, such as allowing players to attack only when their distance from the boss is less than a specific value, so that we will choose more appropriate actions to accelerate convergence during random exploration.

4. Experiments

We evaluated our model based on the following data: clearance rate, which represents the success rate of clearing the boss and completing the game; total average of life, which refers to the average remaining lives, with non-completed attempts recorded as 0; average life of pass, which refers to the remaining lives at the time of completion and average time of pass, which measures the time taken to successfully complete the game.

Furthermore, we found that in high-difficulty battles, it becomes extremely challenging for our model to successfully complete the boss fight. In such cases, clearance rate and average of life may lose their significance as meaningful metrics. Therefore, we adopt the boss's remaining health as a reference indicator in those situations, considering the progress made in depleting the boss's health as a measure of success. This becomes even more evident in our other game, Ori and the Will of the Wisps.

4.1. Basic Experiment of Hollow Knight

We trained our model on the boss Hornet using both Resnet18 and Resnet34 architectures, with approximately 600 attempts for each. Table. 1 displays some basic data from the basic experiment of Hollow Knight.

	Resnet18	Resnet34
Clearance Rate	18.22%	22.49%
Total Average of Life	0.28	0.3
Average Life of Pass	1.53	1.33
Average time of Pass	57.12s	62.33s

Table 1. Some basic data from the basic experiment of Hollow Knight.

Setting aside the in-game operations, we have discovered that our two models have achieved performance levels that are quite close to those of novice human players in these fundamental metrics. And the Resnet34 model has more great performance in both quantitative results, which has been shown in Table. 1 and qualitative results, which was reflected in the gameplay process. This actions performed by the Resnet34 model are smoother and more human-like in nature. Thus, we decide to choose Resnet34 model to further conduct the subsequent researches.

4.2. Operation Selection

As mentioned in the previous section, in order to avoid completely random exploration of the model and accelerate convergence speed, we implement the Operation Selection part in the random action of ϵ-greedy algorithm. In this section, we evaluate the efficiency of different Operation Selection policy. Fig. 3 shows the final results of this part.

4.2.1 Avoid Useless Attack

Among all the actions in our model, the ”attack” action holds significant importance as it is the most commonly used action for defeating bosses and progressing in the game. Additionally, the total number of available actions is not extensive. However, during the mid-training phase, we observed that our model placed excessive weight on the ”attack” action, leading to frequent attacks towards empty spaces. Although this behavior has minimal impact on the overall progress, it reduces efficiency and affects aesthetics, making the model appear less human-like. Consequently, we aim to mitigate these ineffective attacks and reduce their occurrence.

Our approach is relatively simple. The idea is that when our player is too far away from the boss, we prohibit the agent from using the ”attack” action. This is because when the distance is too large, attacking would be ineffective, making it more reasonable to refrain from using the ”attack” action.

4.2.2 More Spells

In Hollow Knight, spells serve as another means of attacking bosses. Players can cast spells by consuming Soul. In this study, we only consider two specific spells: Abyss Shriek and Descending Dark, which is displayed in Fig. 4, referred to as SpellUp and SpellDown, respectively, in the results section. This nomenclature is derived from their respective keyboard inputs (up and down keys) for the spells. Compared to regular attacks, spells have lower frequency of use due to the limitation imposed by available Soul. However, they offer greater flexibility in defeating bosses. As mentioned earlier, our agent tends to favor attacks over spells. Therefore, we also attempt to modify the Operation Selection to encourage the agent to use spells more frequently.

Our approach is to make the agent behave more like a human in using skills. For example, when the boss is directly above the agent, we increase the reward for using Abyss Shriek. This encourages the agent to prioritize the appropriate skill usage in a manner more similar to human decision-making.

4.3. Validation of Hollow Knight

Our validation primarily focuses on the verification of robustness, which involves imposing certain restrictions on the model and assessing its performance under such constraints. These restrictions include acceleration and ban on specific operation.

4.3.1 Accelerate

As the name suggests, acceleration refers to playing the game at a rate multiplied by x. We utilize Cheat Engine to accelerate the game, and then run the pretrained model under normal conditions to evaluate its performance.

Fig. 5 shows the decrease in clearance rate as the acceleration rate increases. The rate of decrease exhibits a close-to-linear curvature, indicating that its performance in terms of acceleration is somewhat subpar.

4.3.2 Ban Operation

We have a wide range of actions, and we want to evaluate their performance under the condition of disabling some of them. However, we need to ensure that the agent still has the opportunity to kill the boss. Therefore, we retain the fundamental attack action and instead attempt to disable spells and dashes to evaluate its robustness.

Table. 2 presents the basic data obtained from disabling specific actions. It can be observed that our model demonstrates strong robustness when certain actions are disabled. This is primarily because our model prioritizes fundamental attacks and does not heavily rely on other actions.

	Ban Spells	Ban rushes
Clearance Rate	21.87%	19.35%
Total Average of Life	0.31	0.25
Average Life of Pass	1.42	1.33
Average time of Pass	54.34s	52.29s

Table 2. Ban Certain Operations

5. Supplementary Experiment

We then use a similar DQN structure for another popular metroidvania game - Ori and the Will of the Wisps, which has a similar operation system as the Hollow Knight.

5.1. Structure

Unlike Hollow Knight, Ori and the Will of the Wisps has a more flexible view according to the distance between the character and the boss. When the distance is far enough, the player’s view would be pulled away to ensure both the boss and character are on the screen. So ResNet did not seem to be a good idea for Ori and the Will of the Wisps as it is more easily to be overfitted. Moreover, Ori and the Will of the Wisps has a more complex skill system that player could choose to

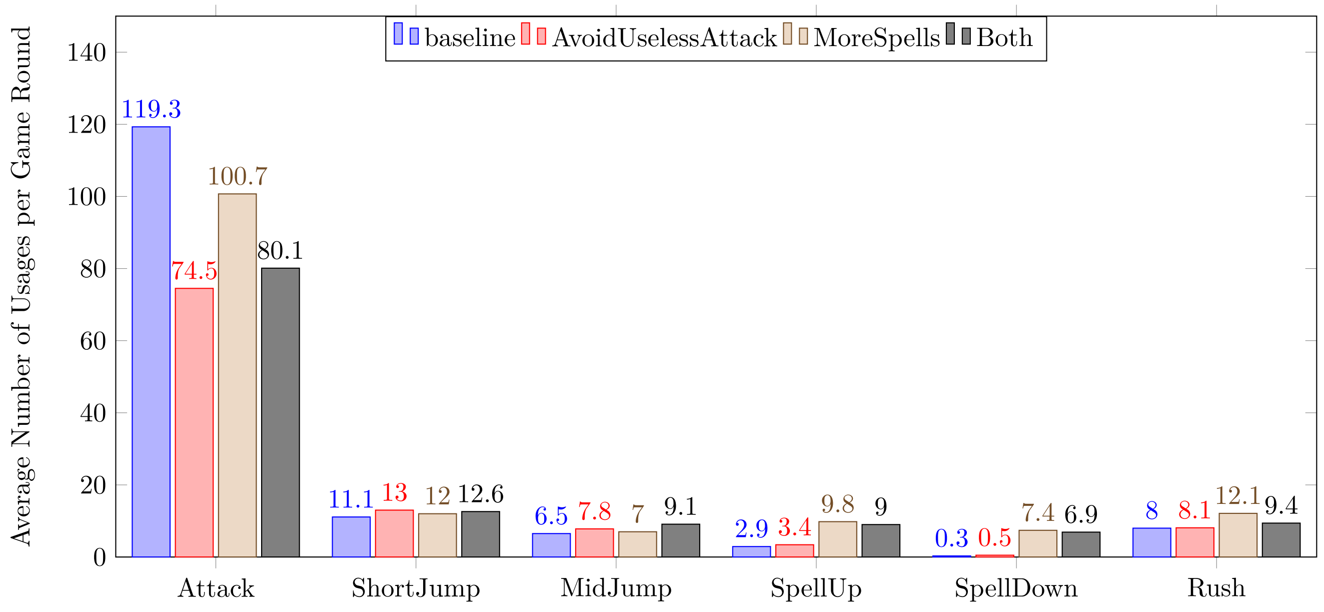

Figure 3. Operation Seletion: This figure presents the average number of actions used per game round under different Operation Selection Policies. It can be observed that the ”AvoidUselessAttack” policy reduces the number of attacks performed by the agent, while the ”MoreSpells” policy encourages the agent to use more skills. The combined use of these policies yields even better results, indicating a more optimal balance between attacks and skills usage.

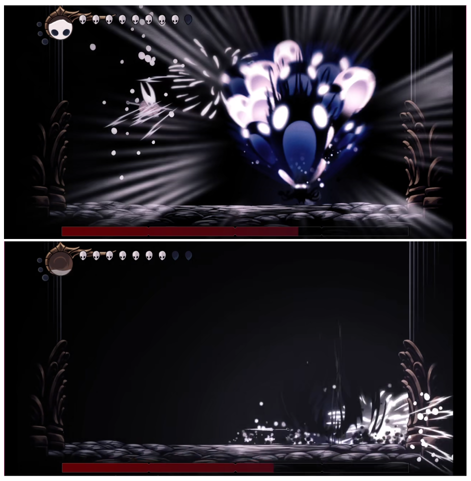

Figure 4. Abyss Shriek (SpellUp) above and Descending Dark (SpellDown) below

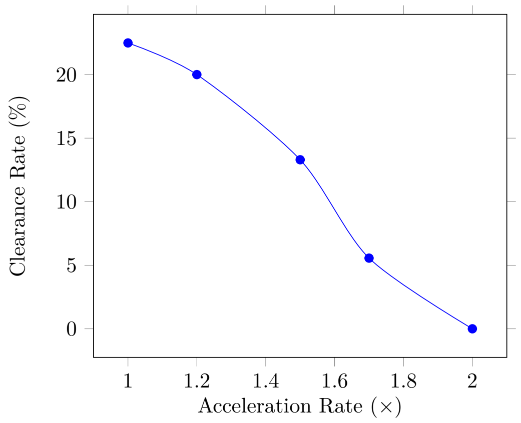

Figure 5. Accelerate: The curve shows the decrease in clearance rate as the acceleration rate increases. And finally it decreases to 0 at 2×.

take different skills in one game. For a quicker learning process and rational comparison with Hollow Knight, we just let the network to learn the basic operations: left, right, rush, attack and jump. Therefore, the DQN just have one brain and fewer convolutional layers, and the pooling layers are also used to prevent overfitting.

5.2. Result

After about 300 attempts for this DQN model, the game AI did not had a good performance in the normal mode for boss rush in Ori and the Will of the Wisps. Table. 3 shows some basic data from the experiment of Ori and the Will of the Wisps.

It is clear that the model did not have a satisfied clearance rate even with the comparison with a novice human player. The alive time is too short as the game

Clearance Rate	Average Life of Boss	Average survival time
5.32%	21.6%	94.7s

Table 3. Some basic data from the experiment of Ori and the Will of the Wisps

AI did not learn how to dodge the attacks efficiently. Also, the special map in the boss rush (when the character move to the left or right wall of this map, the character would also lose some hp) also takes a part in the short survival time. However, the average life of boss showed that the game AI performed well in attacking. Though the boss would only be damaged when the character attacks from the behind, the game AI could seize the opportunities effectively.

5.3. To Do

Based on the characteristic of Ori and the Will of the Wisps's operation system and the change of view, more training time may be a possible way to improve the model performance. As the survival time is too short for the game AI to learn the proper action, and when the character was beaten, it would have a short hit stun and so may not directly operate the next action, the 300 attempts training time seems to be too short for the game AI to learn how to evade the damage. As the model just use the basic operations, some of the skills of the boss can not be evaded. Add some operations of the character that could help the agent to dodge the attack from boss and help it to have a longer survival time is also a possible method to improve the model performance.

6. Future Works

Two aspects of our follow-up work are worth continuing. The first is the improvement and strengthening of the game AI for Hollow Knight. Because the knowledge base of the model is shallow and the training time is little, although this AI model can already reach the level of ordinary beginners, there is still a lot of space for improvement. According to Validation, AI will be huge affected when the input or output is limited, which indicates that our AI model does not have high robustness. Therefore, expanding the knowledge base of the model and increasing the training time will be our first concern. The second is the verification of AI model extensibility. Due to the short time of experiment, this report failed to obtain great training results in Ori and the Will of the Wisps. In the future, the structure and parameters of DQN will be adjusted to obtain better results.

7. Workload Distribution

Zixian Zhang is responsible for training and organizing experimental data for Hollow Knight, conducting evaluations, and attempting to optimize the network structure. Cunhan You is responsible for conceptualizing the basic network structure of Hollow Knight and optimizing the DQN algorithm parameters, such as the composition of the reward, and collecting and integrating data from two games to read the code. Yi Wang is responsible for training and organizing experimental data for Ori and the Will of the Wisps, conducting evaluations and attempting to optimize the network structure. Ruiqian Li is responsible for adjusting the basic network structure of Ori and the Will of the Wisps and report writing.

8. Conclusion

In summary, we have demonstrated through experiments that our AI model can complete some of the Boss fights of the Hollow Knight. The results show that the clearance rate of this AI can reach 22.49%. Research has found that expanding the knowledge base of AI models can effectively accelerate the training process and effectiveness. Due to the fact that the majority of its learning background comes from the boss fights itself, it has flexibility and scalability. Future work will further refine this AI and conduct more scalability experiments.

References

[1] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013. 2

[2] Harrison Ho, Varun Ramesh, and Eduardo Torres Montano. Neuralkart - a real-time mario kart ai using cnns, offline search, and dagger. https://github.com/rameshvarun/NeuralKart, 2021. 2

[3] Junjie Li, Sotetsu Koyamada, Qiwei Ye, Guoqing Liu, Chao Wang, Ruihan Yang, Li Zhao, Tao Qin, Tie-Yan Liu, and Hsiao-Wuen Hon. Suphx: Mastering mahjong with deep reinforcement learning. *arXiv preprint arXiv:2003.13590*, 2020. 2

[4] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 2