Generalizing a deep q-learning agent to environments with the same action space in Atari shooting games

Maryam Alghfeli
Maryam.Alghfeli@mbzuai.ac.ae

Mario Cantero
Mario.Cantero@mbzuai.ac.ae

Nouf Alshamsi
Nouf.Alshamsi@mbzuai.ac.ae

1 Introduction

1.1 Project overview

Machine learning is about analyzing data to make decisions or predictions. It is usually categorized as supervised, unsupervised, and reinforcement learning. In supervised learning the main objective is to find an optimal, generalized model for prediction from labeled data. In unsupervised learning, the machine is trained using an unlabeled dataset, and the main objective is to learn similarities, differences, and patterns. Reinforcement learning is about building an agent that learns from actions by interacting with the environment through trial and error to receive rewards as feedback to solve control tasks or decision making problems. This approach tries to learn an optimal policy and takes action for a given state to maximize the rewards through training [1].

The integration of reinforcement learning and deep learning gives a wide range of field applications in games, natural language processing, computer vision, robotics, energy, business management, finance, healthcare, education, etc. The main goal of this project is to connect a reinforcement learning algorithm with a deep neural network (DNN) to create an agent that learns to play shooting games from the Atari environments. These games are implemented in the Arcade Learning Environment (ALE), and most of them share the same or similar action space. An agent will learn from a single environment of a selected set, and then it will be evaluated on the other environments to see if it generalizes. A general view of how a reinforcement learning task works is shown in figure 1.

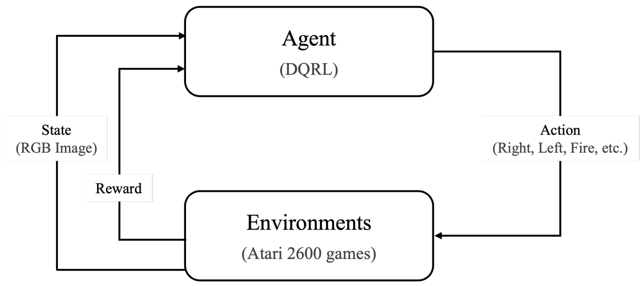

Figure 1: Reinforcement learning cycle

In the implementation of the project, we will use the OpenAI Gym environments and DNN for the agent, which operates directly on RGB images. In this approach, the input is an image at each time step, and the output is the next action that is more likely to maximize a future reward for the agent. Therefore, Atari games require image processing techniques to the frames to learn features and use them to implement algorithms to run the games.

The Deep Neural Network is inspired by the structure of neurons in the human brain. Neural Networks are made up of layers of neurons, which are the core processing unit of the network. It consists of an input layer, hidden layers and an output layer. By feeding the raw input data into a deep neural network (policy), we will get the output which is the probability of each action for maximizing a future reward, hence choosing the one with the highest probability.

OpenAI Gym is an open-source python library for testing different reinforcement learning algorithms on various simulated environments. It works with a variety of frameworks, including Keras and TensorFlow. It is straightforward to grasp, makes no assumptions about the agent's structure and provides an interface to all reinforcement learning tasks [2].

1.2 Related work

Mnih et al. in [3] propose a deep learning model that follows a reinforcement learning approach to learn control policies from raw pixels as input. Convolutional Neural Network (CNN) model was used to output a value function to predict future rewards. A single neural network agent trained using variant Q-learning algorithms. This method was applied to a range of Atari 2600 games from the Arcade Learning Environment. This approach produces cutting-edge outcomes in six of the seven games it was tested on, with no modifications to the architecture or hyperparameters.

Moreover, Chen et al. in [4] proposed in a course project to add of a recurrent neural network on top of the conventional DQN architecture with the idea of adding importance to information learned on several previous time steps. This idea came from the fact that there are some games where past information is relevant to continue successfully in the game. They trained four different policy architectures in two different games: a conventional DQN, a conventional deep recurrent neural network (DRQN), a DRQN with linear attention, and finally, another DRQN with global attention. In the end, the conventional DRQN performed better than the conventional DQN in both Q*bert and Seaquest games. Although they did not use Atari shooting games, the idea of using a more complex deep neural network architecture on top of a simple one can make a significant difference in the performance of the agent, and we can leverage that flexibility if we have time for improvement in this month, and depending on how our DQN agent performs.

2 Motivation

We already know that reinforcement learning can be useful in real-life applications like healthcare, energy, transportation, or robotics, but in this case, we are leveraging the simplicity of a video game application to build a solid foundation on this paradigm to later work on the former situations on a research level. In addition, we chose this application because the intuition of how the algorithm works in terms of rewards and actions is more friendly to grasp in a video game.

3 Evaluation

There are multiple approaches to evaluate a reinforcement learning algorithm, and the most suitable way might depend on the objective of the project. In this case, we plan to train an agent with deep q-learning on a shooting game and check its performance on other shooting games to test generalization. First, in order to know when an agent is done learning, we will set a hyperparameter ε that indicates the threshold of the difference in average reward between a current episode n and a previous episode $n-1$. If the error between the current and previous episode is less than ε, then we can consider that the agent has finished its training phase. A clear visualization of this is done by plotting the episodes versus the average reward per episode, like the figure 2 that was produced by Rodrigo Cesar Bonini et al. [5] between a Q-learning algorithm and their proposed algorithm called PCO.

After training the agent, we test the algorithm on the other games by running several episodes and compare the average of the final score with every other game. At the end, we will have a g by g matrix where rows of which will represent the games where an agent was trained on and the columns will represent the games the agent was tested on, filled with the mean of final reward in k episodes for each game. The diagonal of the matrix will correspond to an agent tested on its own training environment, which in this case it would not be useful if we are aiming generalization.

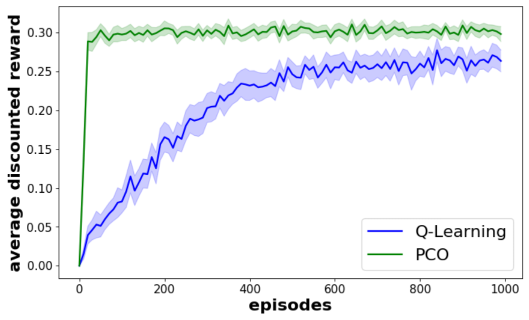

Figure 2: Example of an episode vs average reward plot

4 Resources

The dataset in reinforcement learning is obtained by the agent's interaction with the environment; the agent experience is considered the training data, and there is no data collection step. However, this is done in real-time with continuous feedback, which requires millions of time training steps as the agent receives a state (image frame) from the environment, and based on the state, the agent takes action. Then, the environment goes to a new state (new image frame), and the environment provides the reward to an agent, and the goal is to maximize the cumulative reward. Therefore, it requires a lot of computational resources.

One of the classical reinforcement learning environments is the OpenAI Gym, which supports training agents in different branches, including the Atari games. OpenAI gym has a wide range of environments and supports the ease of implementing custom environments to work on reinforcement algorithms. We will use a programming environment that supports python, such as Google Colab, Jupyter Notebook, etc., for the OpenAI Gym with Pytorch framework to build the DNN. Moreover, an RTX 3060 could be used as a hardware-supporting GPU or lab computer if we need more computational power.

References

[1] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[2] Sudharsan Ravichandiran. *Hands-on reinforcement learning with Python: master reinforcement and deep reinforcement learning using OpenAI gym and TensorFlow*. Packt Publishing Ltd, 2018.

[3] 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.

[4] Clare Chen, Vincent Ying, and Dillon Laird. Deep q-learning with recurrent neural networks. 2016.

[5] Rodrigo Cesar Bonini, Felipe Leno da Silva, Ruben Glatt, and Anna Helena Reali Costa. Transferring probabilistic options in reinforcement learning. *16th Conference on Autonomous Agents and MultiAgent Systems*, 2017.