CS 464 – Progress Report
Breakout Atari™ Game with Reinforcement Learning

Group 4
Abdullah Arda Aşçı (21702748), Alim Toprak Fırat (21600587),
Atahan Yorgancı (21702349), Tuna Alikaşifoğlu (21702125)

I. Introduction

In our term project we have decided on using reinforcement learning algorithms to learn how to play Atari™ Breakout game. Since our project involves reinforcement learning, we will not employ any labeled or non-labeled datasets.

In Breakout, the player controls a paddle at the bottom of the screen to hit a single ball at the bricks above in a manner similar to "pong", and upon hitting bricks the player scores.

II. Background Information

Instead, the actual learning will occur through the interaction between the learning agent and the game environment which is chosen to be the old Atari™ game Breakout.

For our environment we will be tuning the reward associated with scoring, and missing the ball to encourage RL agent to learn about the game.

Fig. 1. Sample Atari™ Breakout Game with Random Agent [1]

As previously mentioned for our term project, we wish to design and train a deep neural network that can play the nostalgic Atari™ Breakout game. The agent is expected to have an "understanding" of game mechanics such as how to score, and what moves to avoid. Essentially the agent is expected to have a functioning Atari™ skills. A set of rewards and punishments, which are associated with certain in-game actions and outcomes, will be used to train the model who play the game over and over again. At each epoch the artificial intelligence should be a better "gamer" than the previous one.

We are planning to conduct a research about reinforcement learning, deep neural networks and more specific algorithms like Deep Q-networks (DQN) that combines Q-learning with deep neural networks to let reinforcement learning to be applied to complex, high dimensional environments, like video games [2]. Furthermore, we are planning to obtain a working, playable version of the Breakout game which is suitable for training and testing our artificial agent.

III. Work Done & Reaming Work

A. *Work Done*

In our proposal, we stated that we would conduct a research on reinforcement learning, deep neural networks and more specific algorithms like Deep Q-networks (DQN) that combines Q-learning with deep neural networks to let reinforcement learning to be applied to complex, high dimensional environments, like video games [2]. We conducted the promised research to find the optimal approaches to the problem in hand. We informed ourselves on the basics of reinforcement learning and especially on Q-learning with DQN implementations. There are more recent solutions like DeepMind's *Agent57* [3] that can outperform DQN based solutions for atari games. However, these type of approaches are necessary for games that have a credit assignment problem, which means that if we choose an action, and we only win or lose hundreds of actions later, leaving us with no idea as to which of our actions led to this win or loss, making it difficult to learn from our actions. The chosen Breakout game is not that complex nor that far-fetched. Thus, our prior research concludes that DQN based solution will be more than sufficient to outperform the atari human benchmark for the Breakout game.

In addition to our background research, we declared that we would obtain a working, playable version of the Breakout game which is suitable for training and testing our artificial agent. In this context, we are utilizing *OpenAI*'s "Gym" environments for the implementation of the Breakout atari game. Basically there are two versions of the Breakout game present as a Gym environment. One of the versions provides information of the RAM throughout the game [1], and the other provides 3-channel image for each frame throughout the game [4]. Currently both versions of the Breakout game can be initiated from our implementation, and we will decide which one of these versions is the optimum for our case after some experimentation.

At this point of the project, we obtained a playable version of the Breakout game with an human agent, that we call the "keyboard agent" by utilizing the example provided in the *GitHub* repo of `openai/gym` [5]. With this aspect of the program, an human agent can play the Breakout atari game by using the keyboard. Version of the game, either RAM based or image based, and the game speed can be adjusted using CLI arguments. In addition to our playable version, we also managed to initialize an artificial agent based version which is suitable for training and testing. Currently, this agent only takes random actions from the action space. However, this environment is crucial for the remaining of the project, and the training and testing will be conducted on this environment, details are provided in the next section.

B. Remaining Work

At this point we have both RAM and image based Breakout atari game, with an artificial agent and keyboard controlled versions. Currently, artificial agent only takes random actions, but we are planning to thrive on obtaining an artificial agent that outperforms human benchmark. The environments for the reinforcement learning aspect of the project are built, and they are operating successfully. The remaining work is to generate DQN based solution to improve the performance of the artificial agent. In the light of our background research, we believe we can provide an elegant approach. Finally, our research unveils that we should obtain an artificial agent that meets our requirements approximately within 70-96 hours of training.

IV. Work Breakdown Structure

References

[1] OpenAI. (Oct. 2019). "Gym," [Online]. Available: https://gym.openai.com/envs/Breakout-ram-v0/. [Accessed: Mar. 3, 2021].

[2] ——, (Sep. 2020). "Openai baselines: Dqn," [Online]. Available: https://openai.com/blog/openai-baselines-dqn/. [Accessed: Mar. 3, 2021].

[3] A. P. Badia, B. Piot, S. Kapturowski, P. Sprechmann, A. Vitvitskyi, D. Guo, and C. Blundell, *Agent57: Outperforming the atari human benchmark*, 2020. arXiv: 2003.13350 [cs.LG].

[4] OpenAI. (Oct. 2019). "Gym," [Online]. Available: https://gym.openai.com/envs/Breakout-v0/. [Accessed: Mar. 3, 2021].

[5] Openai, *Openai/gym*, Apr. 2020. [Online]. Available: https://github.com/openai/gym/blob/master/examples/agents/keyboard_agent.py.

Appendix A
Python Code

```python
import gym
import time
import click


@click.group()
def cli():
    pass


@cli.command()
@click.option('--env_name', default='Breakout-v0',
              help='Name of the gym environment.')
@click.option('--it_count', default=0,
              help='Bound number of ')
@click.option('--delay', default=0.0,
              help='Delay duration to set the game speed.')
def random_agent(env_name, it_count, delay):
    env = gym.make(env_name)
    observation = env.reset()

    iteration = 0
    while iteration <= it_count:
        env.render()
        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        if done:
            observation = env.reset()
        if it_count > 0:
            iteration += 1
        time.sleep(delay)
    env.close()


@cli.command()
@click.option('--env_name', default='Breakout-v0',
              help='Name of the gym environment.')
@click.option('--delay', default=0.1,
              help='Delay duration to set the game speed.')
def keyboard_agent(env_name, delay):
    env = gym.make(env_name)
    ACTIONS = env.action_space.n
    SKIP_CONTROL = 0

    def key_press(key, mod):
        global human_agent_action, \
            human_wants_restart, \
            human_sets_pause
        if key == 0xff0d:
            human_wants_restart = True
        if key == 32:
            human_sets_pause = not human_sets_pause
        a = int(key - ord('0'))
        if a <= 0 or a >= ACTIONS:
            return
        human_agent_action = a

    def key_release(key, mod):
        global human_agent_action
        a = int(key - ord('0'))
        if a <= 0 or a >= ACTIONS:
            return
        if human_agent_action == a:
            human_agent_action = 0

    env.render()
    env.unwrapped.viewer.window.on_key_press = key_press
    env.unwrapped.viewer.window.on_key_release = key_release

    def rollout(env):
        global human_agent_action, \
            human_wants_restart, \
            human_sets_pause
        human_wants_restart = False
        obser = env.reset()
        skip = 0
        total_reward = 0
        total_timesteps = 0
        while True:
            if not skip:
                a = human_agent_action
                total_timesteps += 1
                skip = SKIP_CONTROL
            else:
                skip -= 1

            obser, r, done, info = env.step(a)
            if r != 0:
                print('reward %0.3f' % r)
            total_reward += r
            window_still_open = env.render()
            if not window_still_open:
                return False
            if done or human_wants_restart:
                break
            while human_sets_pause:
                env.render()
                time.sleep(delay)
            time.sleep(delay)
        print((f'timesteps {total_timesteps} '
               f'reward {total_reward:0.2f}'))

    print(f'ACTIONS={ACTIONS}')
    print('Press keys 1 2 3 ... to take actions 1 2 3 ...')
    print('No keys pressed is taking action 0')

    while True:
        window_still_open = rollout(env)
        if window_still_open == False:
            break


if __name__ == '__main__':
    # Keyboard Global Attributes
    human_agent_action = 0
    human_wants_restart = False
    human_sets_pause = False
    cli()
```