Develop AI agent for Atari game

Project Members - Group 105

Roi Machluf 204670889 roi.machluf@mail.huji.ac.il

Almog Mor   206123267 almog.mor@mail.huji.ac.il

Supervisor

Dr. Refael Vivanti, RAFAELvivanti@gmail.com
Rafael-Dynamic Defense Company

Mentor

Mr Leshem Choshen, Leshem.choshen@mail.huji.ac.il

Abstract

Learning to control agents based on environment sensors is one of the challenges which rise when developing policies for complex tasks. Achieving superhuman performance in Atari games is a good indicator that a policy will perform well in such a task.

Atari's Skiing game is a classic driving challenge which requires a set of skills such as navigation, exploration and control. Hence, we define the problem of the project to be developing an AI agent for the Atari's Skiing game.

We chose to solve the problem using Reinforcement learning. A Reinforcement Learning agent interacts with its environment in discrete time steps. In each iteration, the agent receives an observation and a reward and chooses a legal action. Next, the environment updates to a new state and rewards the agent accordingly, and the cycle repeats.

We used the DQN algorithm, where the agent uses a deep Q-network to approximate the Q-value function for each action. The policy determines the next action by the maximum output of the Q-network. Our main challenges were finding the best neural network architecture, tuning the parameters of the model, and developing features which help the network to converge.

The goal of Skiing is to reach a score closest to zero, the score descends as the time runs. Our up-to-date agent's score is -4837, Third place after Go-Explore (-3660) and agent57 (-4202.6). The best score achieved by a human player is -3686.

Introduction

The goal of our project is to develop an AI agent for the skiing Atari game.

Skiing is a single player game. The player uses the joystick to control the direction and speed of a stationary skier at the top of the screen. He must avoid obstacles like trees and moguls. The goal of the game is to run downhill through all gates as fast as possible. Each missed gate results in a five-second penalty. There is a reward at the end that is proportional to the time spent.

Trained agent which is capable of sensing its environment and moving safely without human input is a solution to many day to day applications such as autonomous vehicles, autonomous drones etc. Therefore learning the affection of different features and methods and success in finding the good combination of them, can help us in training such agents. This solution is in the interest of many high tech companies, in particular 'Refael' which study this field.

The main challenges of this problem are:

<u>long term credit assignment</u>: which decisions are most deserving of credit for the positive (or negative) outcome that follow? This problem is particularly hard in skiing, because rewards are delayed and credit needs to be assigned over long sequences or actions.

<u>The control:</u> the control is by the derivative. Each steering action changes the direction of the skis, and the skis control the change in the location. Therefore the location is the second derivative of the action.

We assume that these challenges are the reason that a human player held the best score record for this game when we chose it although there were several attempts to over perform his score, including 'Agent57' by Deepmind.

Our approach for the solution includes classic Reinforcement Learning setup, with a simple Neural Network, manipulated reward and other features which tries to lower the complexity of the problem.

Hence, we will show in this paper that with a simpler Neural Network and less resources than other solutions to this problem, we succeeded in getting a good result which competes with the average human player and getting close to the best performance achieved by a human player.

Related work

At 2013, DeepMind presented a deep learning model[1] which learns to control policies from high-dimensional sensory input of raw pixels. Using this model which is a convolutional neural network, they had successfully outperformed 6 of Atari games compared to other previous models and surpassed a human expert's score on three of them.

In 2015, DeepMind developed another agent, termed a deep Q-network, that can learn policies directly from high-dimensional sensory inputs using reinforcement learning. This agent achieved a level comparable to that of a human across a set of 49 atari games[2].

The University of California, Berkeley published a paper about Trust Region Policy Optimization (TRPO)-A practical algorithm which optimizes large nonlinear policies such as neural networks by making some approximations to the theoretically justified scheme.

Their experiments demonstrate robust performance on several tasks, including playing Atari games using images of the screen as input[3].

The last breakthrough in this area was achieved by Deep Mind at 30.3.2020 with their paper about 'Agent57'[4] which is the first deep RL agent that outperforms the standard human benchmark on all 57 Atari games.

Vivanti et al[5] Use Reward Noising to overcome the Boring Areas Trap and Manipulative Consultant Problems in Reinforcement Learning problems. We tried using the methods suggested.

[1] "Playing Atari with Deep Reinforcement Learning." https://arxiv.org/abs/1312.5602.

[2] "Human-level control through deep reinforcement learning " 25 Feb. 2015, https://www.nature.com/articles/nature14236.

[3]Schulman, John, et al. "Trust region policy optimization." *International conference on machine learning*. 2015. http://proceedings.mlr.press/v37/schulman15.pdf.

[4] "Agent57: Outperforming the Atari Human Benchmark." 30 Mar. 2020, https://arxiv.org/abs/2003.13350.

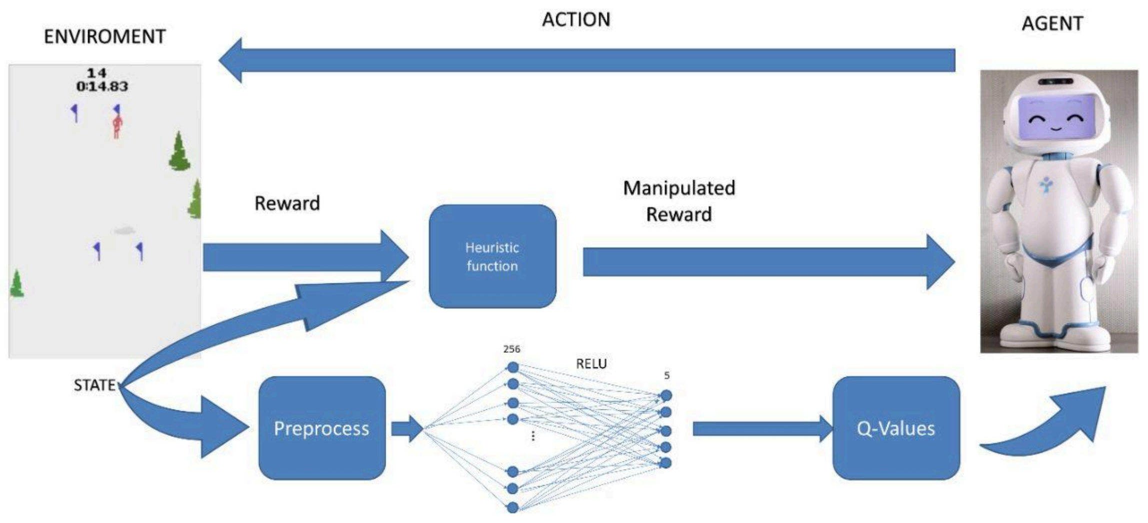
[5] Vivanti, Refael, Sohlberg-Baris Talya D. ,Cohen Shlomo, and Cohen Orna. "Adaptive Symmetric Reward Noising for Reinforcement Learning." *arXiv preprint arXiv:1905.10144* (2019).

Methods & Materials

We chose to solve the problem using Reinforcement learning. A reinforcement learning agent interacts with its environment in discrete time steps. At each time, the agent receives an observation and a reward and chooses a legal action. Next, the environment moves to a new state and a reward accordingly, and the cycle repeats.

We used the Deep Q network algorithm which tries to predict the q-value of action in a particular state. The algorithm is model-free, the input of the NN is the image and it does not require a model of the environment. Using this Q action value function, we can select the action which maximizes the q value.

This diagram represents our solution's pipeline at each discrete time stamp:



Key components:

- **Agent:** The algorithm is referred to as the agent.
- **Environment:** The environment in which the agent operates. In our game, this includes the positions of the player, flags and trees.
- **State:** the current condition of the environment.
- **Action:** the action the agent takes depends on the state - move right, left or straight.
- **Reward:** the agent receives positive or negative rewards depending on the previous actions.

To improve score performance in fewer training episodes, we estimated many features during the development process.
Some of the features included changing the network architecture, adding gaussian noise to the reward as Rafi our advisor suggested in his paper[6] and different preprocessing methods for the image input. From all the features we've tested along the way, the following were identified as improving performance:

- **Expanding the action space:** Every steering action changes the direction of the skis, which control the change in location. Thus, the location is the second derivative of the action. To assist the agent in controlling the steering, we chose some series of actions which make moving much easier.

- **Player in center (Preprocces):** In every frame we preprocess the frame and place the player in the center of the image. So, the player stays still on the screen, but the network will only need to find the flags and trees, so we could simplify the network architecture.

- **Rewards with heuristic:** In Skiing, rewards are delayed and credit needs to be assigned over long sequences or actions. In order to make the reward more immediate we manipulate the reward using a heuristic function which calculates the distance between the player and the closest flags. The manipulated reward is normalized between -1 and 1.

- **Network architecture:** At first we tried several architectures from different papers with similar tasks. Eventually, we used NN with 2 FC layers with Relu activation function.

[6] Vivanti, Refael, Sohlberg-Baris Talya D. ,Cohen Shlomo, and Cohen Orna. "Adaptive Symmetric Reward Noising for Reinforcement Learning." *arXiv preprint arXiv:1905.10144* (2019).

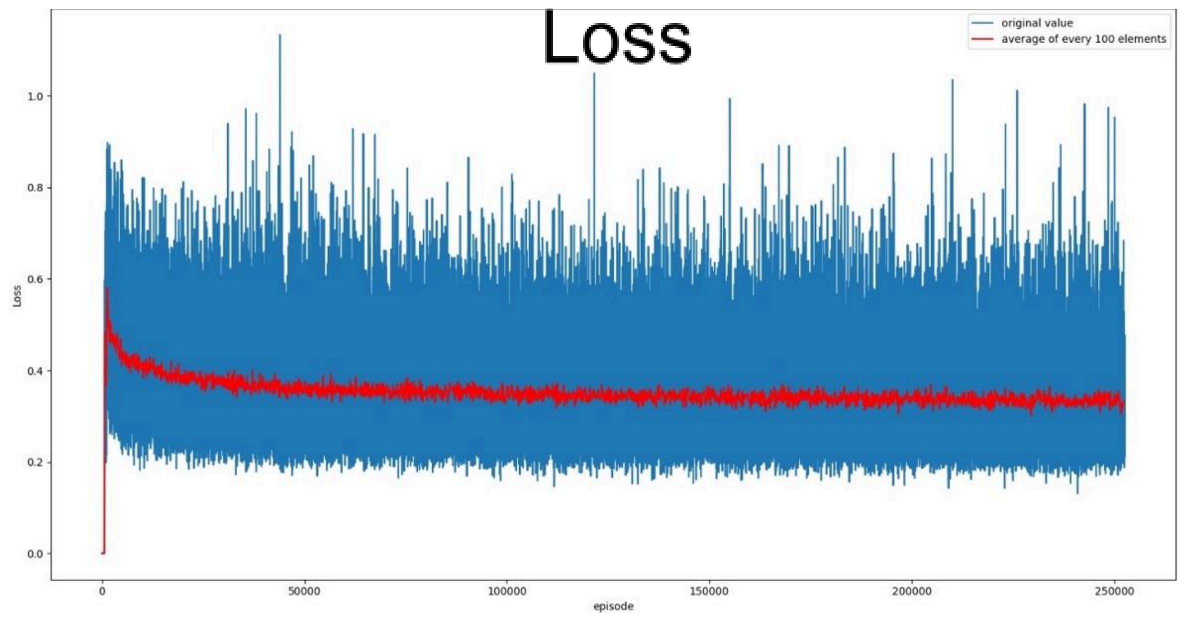
Evaluation

In order to evaluate each version, we built an evaluation tool that plays the game 100 episodes with a given trained agent. This tool calculates the mean of the scores. Whenever we developed a new feature, we compared the new trained agent to a benchmark agent using the evaluation tool.

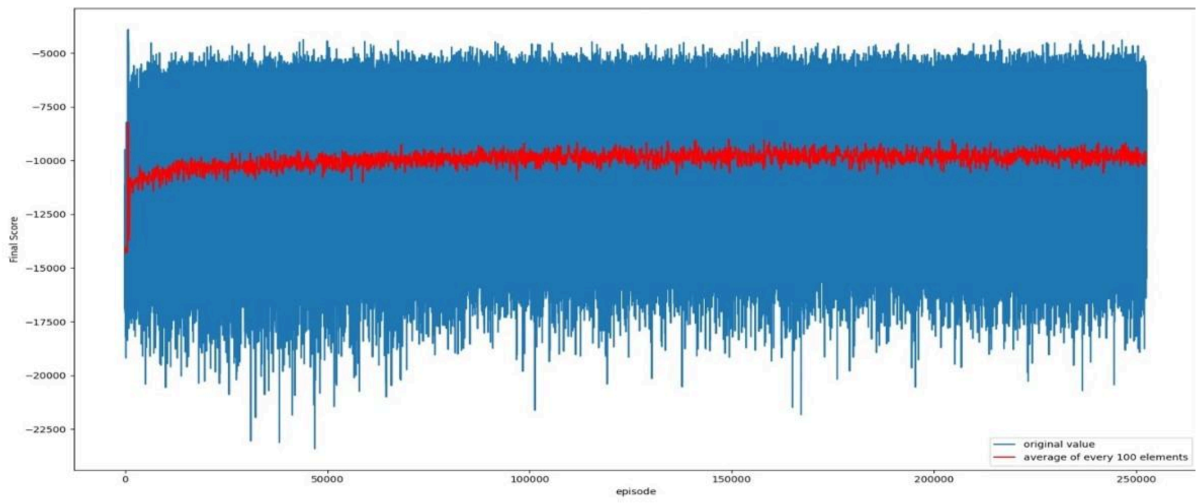
In addition to evaluating an agent, we evaluated the training process by comparing the loss and rewards graphs during the training.

Results

Graphs of losses and rewards are shown below:



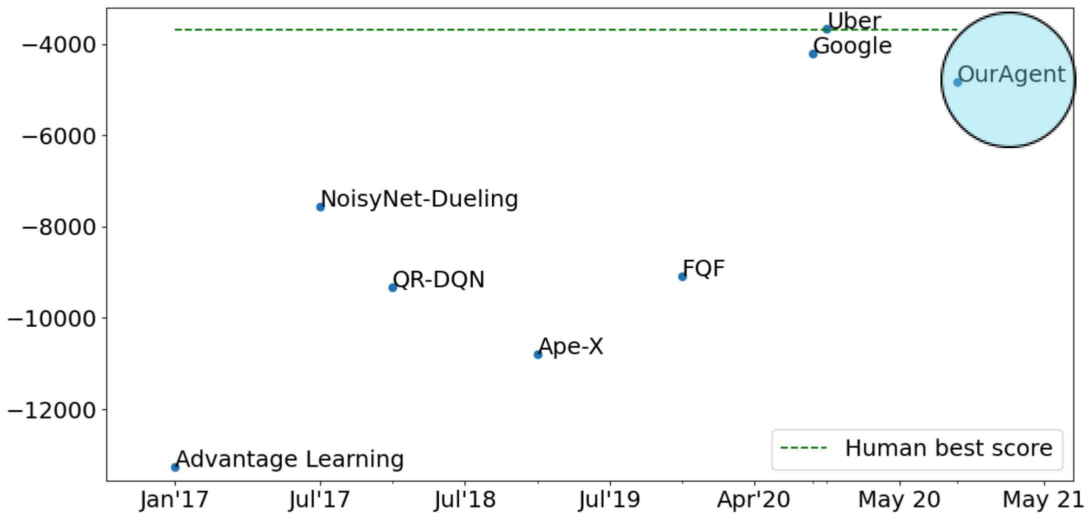
Reward



During the training process, the loss decreases and the reward increases. Due to the epsilon greedy policy, the rewards received during training do not represent the optimal score of the agent.

Our up-to-date agent's score is -4837, while the best score achieved by a human player is -3686.

Many researchers built agents for this challenge. This leaderboard follows these agent's performances. We used these results as a benchmark.



The **x** axis represents the time when the agent was published and the **y** axis shows his result. Our agent is in third place after “Go explore” by Uber AI labs and “Agent 57” by deep mind (Google). The best Human performance is represented as the green line.

Conclusion

Our study showed that a good policy for skiing can be learned using only environment sensors. By decreasing the state space, making the reward more immediate, and preprocessing the input of the network, we were able to use a simple network architecture and aid the network to converge.

Additionally, these methods helped us shorten the training time, which was essential to us due to the limited resources we had. In this way, we could evaluate many ideas and select only the best ones.

The leaderboard reveals that our agent performs well, almost as well as Google and Uber's agents. We have an advantage over the other models in the simplicity of the network architecture.

Future work

As a possible extension to our study, we can use the methods we developed to learn policies for other applications. Other applications include other agents for Atari games, autonomous driving policy, and controlling autonomous drones.

Moreover, our solution may benefit Refael company, which proposed this project.