Rock Paper Scissors Game Using Reinforcement Learning

The classic game of chance

MICROSOFT

30 may 2023

Created by: Massock Batalong M.B.

Rock Paper Scissors Game Using Reinforcement Learning

The classic game of chance

Rock Paper Scissors is a classic game that most of us have played at some point in our lives. It's a game in which two players make simultaneous moves by choosing one of the three options: rock, paper, or scissors. The winner is determined by a set of rules: rock beats scissors, scissors beat paper, and paper beats rock.

Now, let's imagine we want to create an AI agent that can learn to play Rock Paper Scissors competitively. This is where reinforcement learning comes into play.

Reinforcement learning is a machine learning technique that allows an agent to learn from its environment by receiving feedback on its actions. In the case of Rock Paper Scissors, we can create an environment that simulates the game and allows an agent to interact with it. The agent will learn to play the game by taking actions and receiving rewards or penalties based on its performance.

At the beginning of the training, the agent will take random actions, but as it receives feedback, it will learn the rules of the game and develop a strategy to win. The reward system for Rock Paper Scissors would be something like this: the agent gets a reward of +1 if it wins, a reward of -1 if it loses, and a reward of 0 if it's a tie.

In the training phase, the agent will explore different strategies, but over time, it will converge to an optimal strategy that maximizes its rewards. Once the training is complete, the agent will be able to play Rock Paper Scissors at a high level.

Reinforcement learning is a powerful tool that can be used to create AI agents that can learn complex tasks such as playing games. By creating an environment that allows an agent to interact with and learn from its surroundings, we can create intelligent machines that can compete with humans in tasks that were previously considered impossible for machines to achieve.

FIGURE 1 ROCK-PAPER-SCISSOR

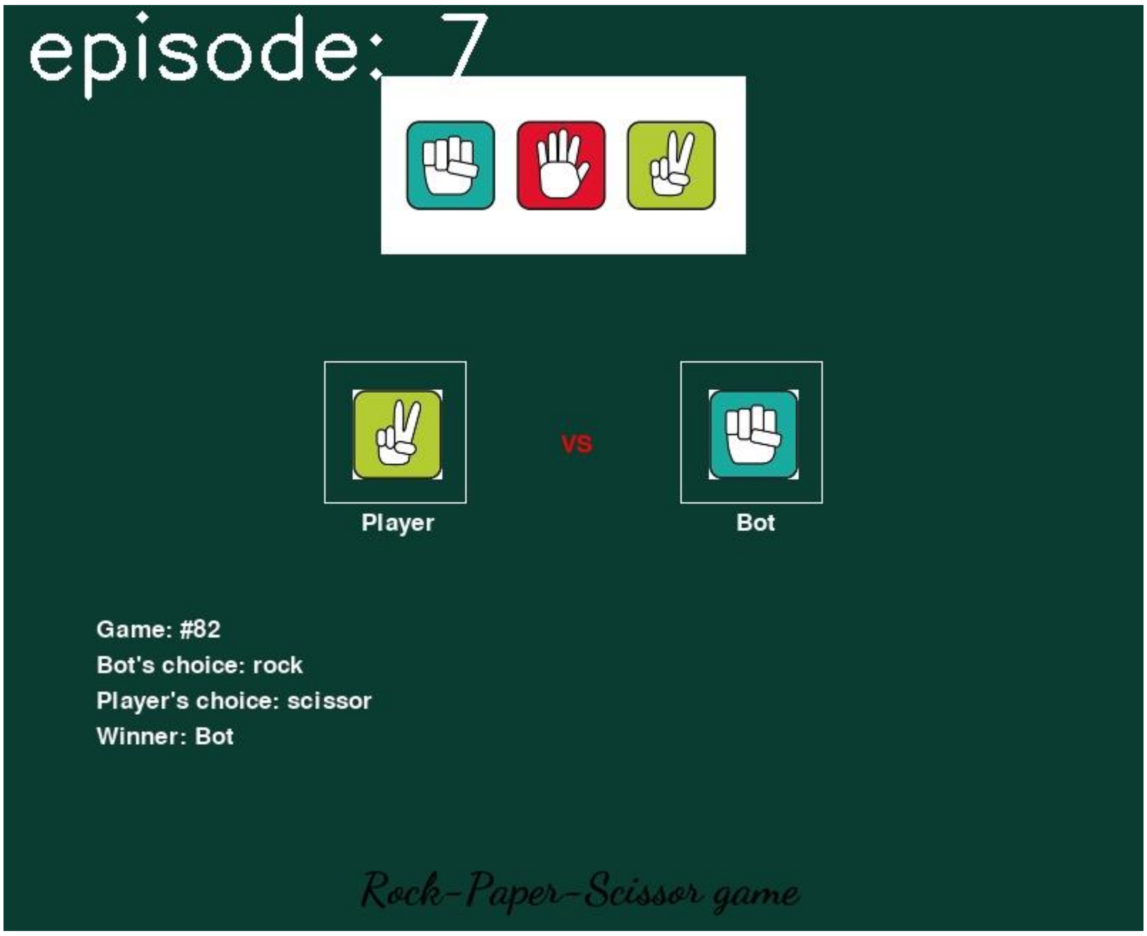
Rock-Paper-Scissors: The Classic Game of Chance

Are you looking for a fun game to play with your friends or family? Then look no further than Rock-Paper-Scissors, the classic game of chance that has been enjoyed by people of all ages for generations.

The Rules:
The game is simple. Each player chooses one of three possible hand gestures: rock, paper, or scissors. *Rock beats scissors, scissors beats paper, and paper beats rock*. If both players choose the same gesture, it's a tie and the game starts over.

Strategy:
While the game may seem entirely based on chance, there are some strategies you can employ to increase your chances of winning. For example, if you notice that your opponent tends to choose rock, you could choose paper to beat them. Or if you think they're going to choose scissors, you might want to choose rock to crush them.

No matter if you add some variations in this game; Rock-Paper-Scissors is a great choice for a simple, fun game that anyone can enjoy. So gather a few friends and let the games begin!



Game: #82
Bot's choice: rock
Player's choice: scissor
Winner: Bot

FIGURE 2 ROCK-PAPER-SCISSOR PLAYS BY AI AGENT

Reinforcement Learning: A Powerful Machine Learning Technique

What is a Reinforcement Learning (RL)?

Reinforcement learning is a machine learning technique that has been gaining increasing attention in recent years. It is a type of deep learning that focuses on providing an agent with the ability to learn and improve based on its interactions with the environment. The agent gets feedback in the form of rewards or punishments, and it uses this feedback to modify its actions in order to maximize rewards and minimize punishments.

The concept of reinforcement learning is closely tied to the idea of trial-and-error learning. In this approach, the agent tries out different actions in the environment and learns which ones lead to positive outcomes. It then uses this knowledge to improve its performance in future interactions.

One of the key features of reinforcement learning is its ability to learn from experience. The agent begins with no knowledge of the environment or the best actions to take, but as it interacts with the environment and receives feedback, it starts to learn which actions are most likely to lead to positive outcomes. This allows the agent to continuously improve its performance over time.

Reinforcement learning is widely used in a variety of applications, including robotics, gaming, finance, and healthcare.

It has shown great promise in solving complex problems that are difficult to solve using traditional machine learning techniques.

Some of the key benefits of reinforcement learning include:

- Ability to learn from experience: Reinforcement learning allows agents to learn from their interactions with the environment, making it ideal for applications where experience is crucial.
- Ability to handle complex problems: Reinforcement learning can handle complex problems that are difficult to solve using traditional machine learning techniques.
- Autonomous learning: Reinforcement learning agents can learn their own without the need for human intervention.

Overall, reinforcement learning is a powerful machine learning technique that has shown great promise in a variety of applications. As the field continues to advance, we can expect to see even more exciting developments in the future.

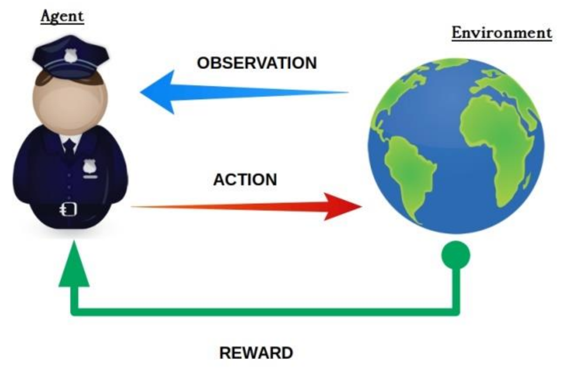


Figure 3 RL environment

Training AI agent to play Rock-Paper-Scissors game

Before to train the AI agent that we have chosen, let's define a term **AI agent**.

What is an AI agent?

An AI agent is a software program that uses artificial intelligence techniques to perform tasks that would normally require human intelligence. It receives input from its environment, makes decisions based on the input using **some *set of rules* or *algorithms***, and generates output that can be used to interact with the environment.
AI agents can be designed for different types of tasks, such **as *playing games, performing automated trading on financial markets, or controlling autonomous robots*.** They can be classified based on various criteria, such as their level of autonomy, their interactions with humans, or the complexity of their decision-making processes. In general, AI agents are designed to be adaptable and able to learn from their experiences, making them useful in a wide range of applications.

Our application, it is playing Rock-Paper-Scissors game, so we have taken four ai agents for training:

1. **DQN (Deep Q-Networks):** This is a deep learning algorithm used to find the optimal action-selection policy for any given environment in a reinforcement learning framework. The algorithm uses a deep neural network to represent the Q-function which maps state-action pairs to their expected discounted cumulative rewards.
2. **PPO (Proximal Policy Optimization):** This is a policy gradient method for reinforcement learning that seeks to optimize the policy directly, rather than the value function. It involves making small steps in policy space and using a trust region to prevent making too large of changes to the policy.
3. **A2C (Advantage Actor-Critic):** This algorithm combines both the actor-critic and advantage functions to improve the stability and performance of the model. The actor-critic architecture has two neural networks running in parallel - the actor network and the critic network.
4. **QRDQN (Quantile Regression Deep Q-Networks):** This algorithm is an extension of DQN, which uses an ensemble of neural networks to estimate the distribution of the Q-value instead of a single value. It has been shown to provide more stable and reliable performance compared to DQN.

In training, the performances of theses ai agents are measured by the RL metrics measures following:

- **Policy loss** is a measure of how well an agent's policy (a set of rules for selecting actions based on current state) performs in a given environment. In other words, it measures the difference between the predicted and actual rewards received by the agent. The goal of reinforcement learning is to minimize the policy loss over time, so that the agent can learn to make better decisions.

$$\mathrm{E}_t[Log\Pi_\theta(a_t|s_t) \cdot A_t]$$

- **Value loss** is a measure of how well an agent can predict future rewards based on its current state. This is done through a value function, which estimates the expected reward for each state. Value loss is the difference between the predicted and actual rewards received by the agent, and the goal is also to minimize it over time, so that the agent can make more accurate predictions about the future rewards it will receive.
- **Policy gradient loss** is a loss function used in reinforcement learning to optimize the policy of an agent. The policy of an agent is a function that takes in the state of the environment and outputs a probability distribution over the possible actions the agent can take.

  The policy gradient loss measures how well the agent's policy is doing at maximizing the expected cumulative reward over time. It does this by computing the gradient of a performance measure, such as the expected cumulative reward, with respect to the policy parameters, and using this gradient to update the policy parameters in a direction that increases the performance measure. The policy gradient loss is typically used in deep reinforcement learning with neural networks to approximate the policy function. The loss function is computed by averaging the product of a set of scores, such as log probabilities or advantages, and their corresponding policy gradients. The result is a scalar value that represents how well the policy is doing and is used to update the policy parameters through backpropagation.

$$\mathrm{E}_{\Pi}[Q^{\Pi}(s,a)\nabla_{\theta} Log\Pi_{\theta}(a|s)]$$

- **Entropy loss** is a term used in reinforcement learning that refers to encouraging an agent to explore its environment by introducing randomness into its actions. In other words, by adding an entropy term to the objective function, the agent is encouraged to take actions that have high uncertainty or a low probability of being chosen. This can help the agent to discover new, potentially more rewarding states, as opposed to getting stuck in a sub-optimal strategy**.**

$$-\sum_{i} P_i \ln(Q_l)$$

- **Explained variance** is a metric used to evaluate the performance of a machine learning model. It measures how much of the variation in the dependent variable (i.e., the target variable) is explained by the independent variable(s) used by the model. In reinforcement learning, explained variance is often used to evaluate how well a value function (i.e., a function that estimates the expected future reward of being in a certain state) is fitting the data. A higher explained variance indicates a better fit, meaning that the value function is accurately predicting the expected future reward.
- **Approximate KL** (Kullback-Leibler) is a term used in machine learning to measure the difference between two probability distributions. It is a common metric used in many machine learning algorithms, including Generative Adversarial Networks (GANs) and reinforcement learning. In reinforcement learning, the KL divergence is used as a metric to measure the difference between the expected and actual policy distributions. The approximation of KL divergence is often necessary because computing the exact value of KL can be difficult and

time-consuming. Therefore, various approximations are used that are faster and easier to compute than the exact value. The most common approximation method is the Monte Carlo estimation, where a large number of random samples are drawn from the distribution and used to estimate the KL divergence.

The learning of each ai agent have taken approximately 4 mins as training duration, This training duration is enough for these agents to know how to play Rock-Paper-Scissor game. Let's see it with tensorboard.

Training results interpretations

A2C training results

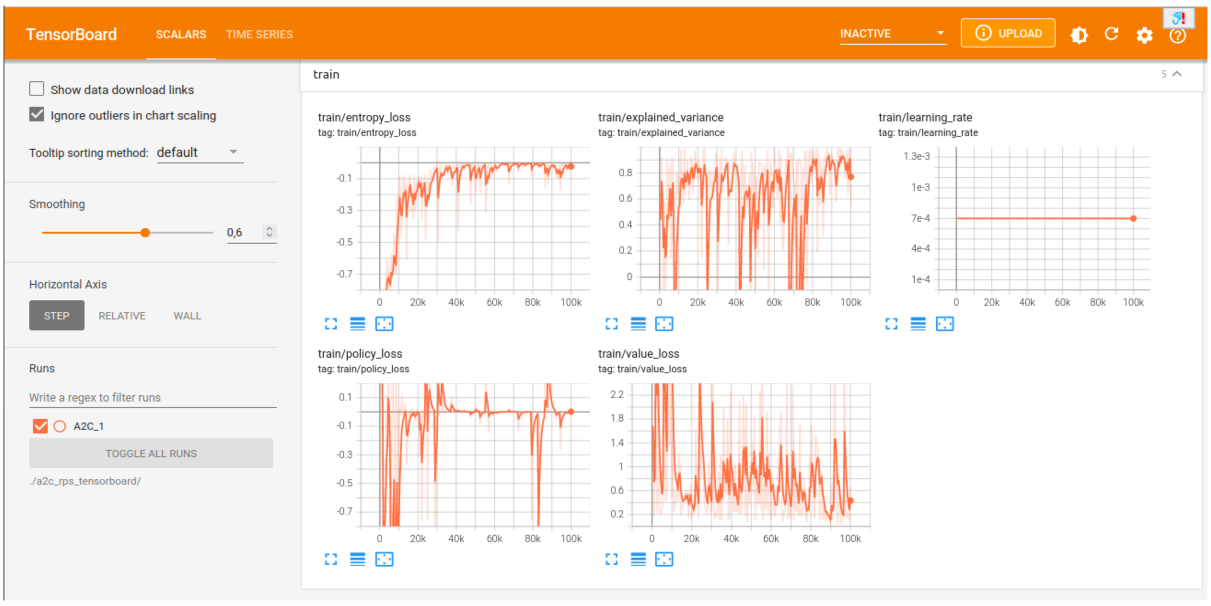

Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing 0,6
Horizontal Axis: STEP
Runs: ☑ A2C_1
./a2c_rps_tensorboard/

FIGURE 4 A2C TENSORBOARD

According to entropy loss and explained variance chart, we see that entropy and variance increases during a training.

Beginning a learning, A2C agent chooses randomly an action. Over time to time that training have passed, a2c agent gets information how it can choose correct action to beat its opponent player. When entropy loss reaches a maximum and this maximum is constant in the remaining time step, agent knows the strategy to win a Rock-Paper-Scissor game.

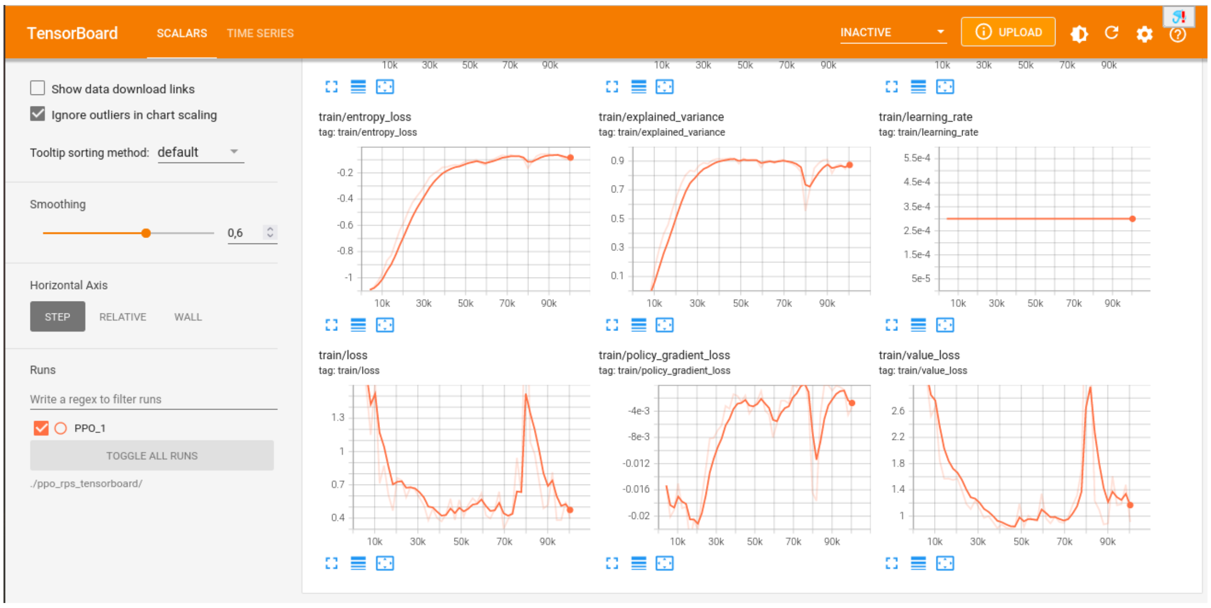
Therefore, the explained variance evaluates a performance of a2c agent; it measures how much of the variation in the target variable (Returns) is explained by the independent variables (*states* $\rightarrow$ *(opponent action, time count, score)*). Over time to time that training evolves, variance increases considerably meaning that the explained variance indicates a better fit of the data; and the value function accurately predicts the expected future reward.

For a value loss, we see how an exponential slope decreases during training. Our a2c agent makes more accurate predictions about the future rewards it receives. On the other hand, policy loss chart said that a2c agent policy performs well in a Rock-Paper-Scissor environment. During training, advantage $\boldsymbol{A_t}$ is sometime positive or negative (see policy loss chart) but it increases in the time. When advantage is positive, policy is also positive meaning that a probability of action chosen is increased.

When it is negative policy loss is also negative (decrease a probability of action chosen).

This four RL metrics measures explains how a2c agent have well learned to play Rock-Paper-Scissor game just only 4 mins.

PPO training results


TensorBoard SCALARS TIME SERIES INACTIVE UPLOAD

- [ ] Show data download links
- [x] Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0,6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

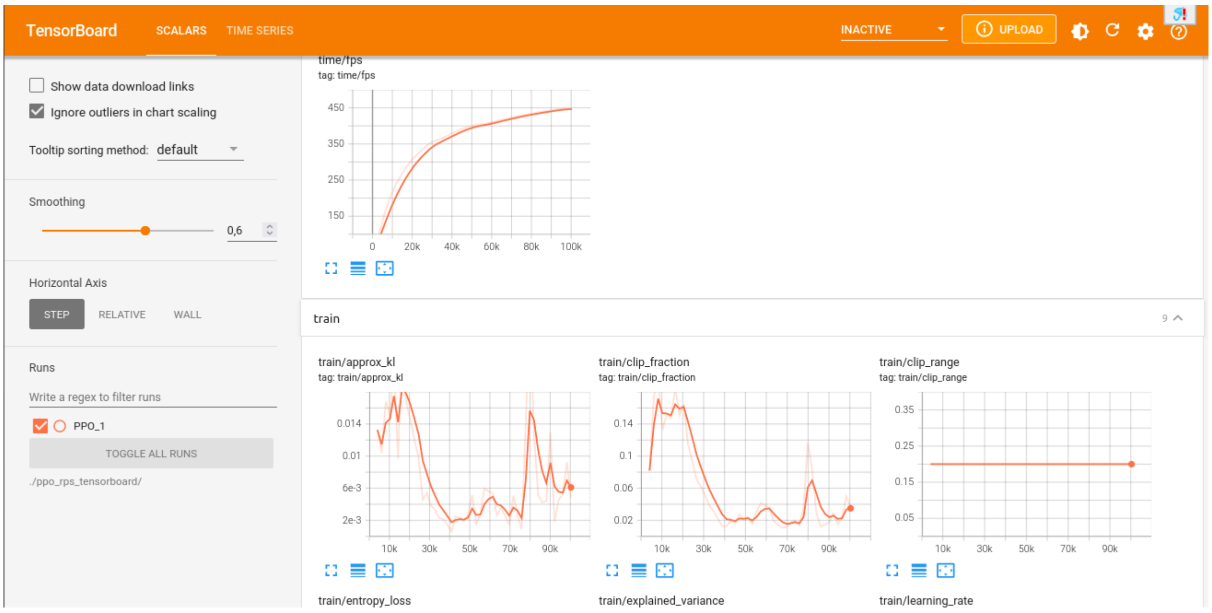
- [x] PPO_1

TOGGLE ALL RUNS

./ppo_rps_tensorboard/

FIGURE 5 PPO TENSORBOARD

PPO agent has same RL metrics measures like a2c agent. We can consider same interpretation. Let's show you another metrics for PPO approx_kl and clip fraction.


TensorBoard SCALARS TIME SERIES INACTIVE UPLOAD

- [ ] Show data download links
- [x] Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0,6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

- [x] PPO_1

TOGGLE ALL RUNS

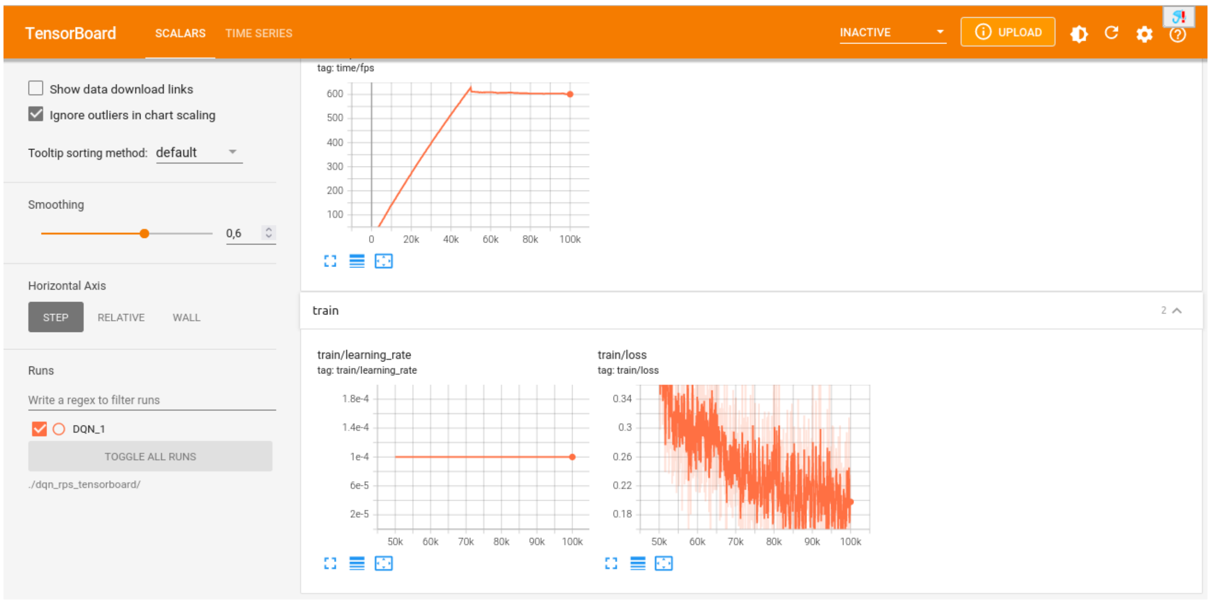
./ppo_rps_tensorboard/

FIGURE 6 PPO TENSORBOARD

Clip fraction is a hyperparameter used in the Proximal Policy Optimization (PPO) algorithm for reinforcement learning. It specifies the maximum change allowed for each parameter update during training. In the PPO algorithm, the objective is to update the policy network so that it can perform better in the environment being trained on. However, if the changes made to the policy network are too large, it can lead to instability in the training process. To prevent this, the clip fraction is used to limit the size of the updates and keep the parameters within a bounded range. Generally, a clip fraction of 0.2 is commonly used in PPO. The value can be adjusted based on the nature of the problem being tackled and the desired performance of the algorithm.

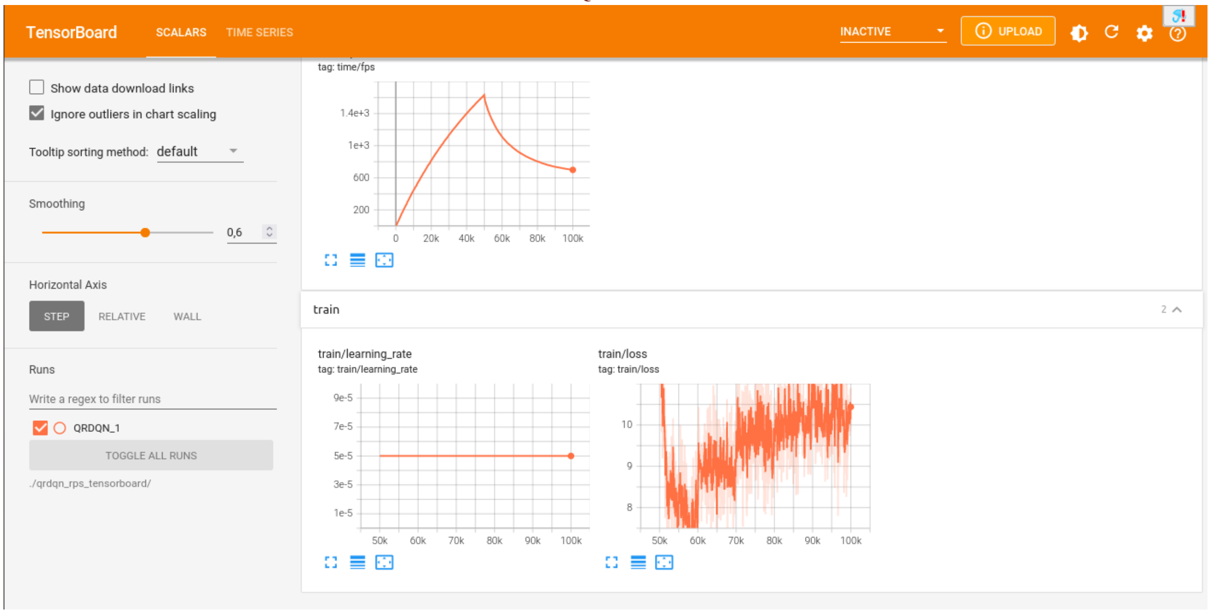
DQN and QRDQN training results

DQN and QRDQN have same RL metrics measures to see how agent performs in the environment. Let's see it.



Show data download links
[✓ Ignore outliers in chart scaling]
Tooltip sorting method: default
Smoothing 0,6
Horizontal Axis [✓ STEP] RELATIVE WALL
Runs
[✓ DQN_1]
./dqn_rps_tensorboard/

FIGURE 7 DQN TENSORBOARD



Show data download links
[✓ Ignore outliers in chart scaling]
Tooltip sorting method: default
Smoothing 0,6
Horizontal Axis [✓ STEP] RELATIVE WALL
Runs
[✓ QRDQN_1]
./qrdqn_rps_tensorboard/

FIGURE 8 QRDQN TENSORBOARD

Let's define a loss function.

A loss function is a mathematical function that measures the difference between the predicted value and the actual value of an outcome variable in a machine learning model.

Agents performances and comparisons

All the four agents have trained against this *statistical agent bot* code below:

```
action_histogram = {}
def statistical(lastOpponentAction, step):
        global action_histogram
        if step == 0:
                action_histogram = {}
        action = lastOpponentAction
        if action not in action_histogram:
                action_histogram[action] = 0
        action_histogram[action] += 1
        mode_action = None
        mode_action_count = None
        for k, v in action_histogram.items():
                if mode_action_count is None or v > mode_action_count:
                        mode_action = k
                        mode_action_count = v
                        continue
        return (mode_action + 1) % 3
```

In each step, this bot receives a last action of an AI agent after it returns its action according to the remainder of Euclidian division by 3. When step is equal to zero, it empties an action_histogram variable. After 4 mins training, we test each ai agent to play against statistical agent in 50 episodes games, we obtain this result:

Table 1 Agents performances

vs	Statistical agent		
agent	win	loose	draw
DQN	46	0	4
QRDQN	46	0	4
PPO	50	0	0
A2C	45	0	5

This tableau shows that PPO is an ai agent that has learned very well how to play Rock-Paper-Scissor game. What happens if we do some tournament between these four ai agents? Let's do it and see the best agent for this game.

Rock-Paper-Scissor leadership

Okay, the tournament consists to play each agent against other agent in the 1st leg and 2nd leg. The agent can be home or away. If an agent plays at home we call *home agent* and if it plays at away we call *away agent*.

Let's go to tournament. Each agent has 6 games; the tournament table result is as follows:

	home_ppo	**home_dqn**	**home_a2c**	**home_qrdqn**
away_ppo		home wins	home wins	home wins
away_dqn	home looses		home looses	home looses
away_a2c	home looses	home wins		home looses
away_qrdqn	home looses	home looses	home looses	

Table 2 rock paper scissor tournament

During the six games,

1. PPO agent has lost all the 3 games in the 1st leg and also the 3 games in the 2nd leg that leads to this result**: 0 win; 6 loose; 0 draw.**
2. DQN agent has won all its 3 games in the 2nd leg that leads to this result: **5 wins; 1 loose; 0 draw.**
3. A2C agent has as result**: 3 wins; 3 loose; 0 draw.**
4. QRDQN has won all its 3 games in the 2nd leg that leads to this result: **4 wins; 2 looses; 0 draw**

The leaderboard table can be made as follows:

Leaderboard on Rock-Paper-Scissor game		
rang	AI agent	Pts
1	DQN	15
2	QRDQN	12
3	A2C	9
4	PPO	0

TABLE 3 AGENTS LEADERBOARD

The following quotation derives from the result obtained from the table above:

It is still not the one who has learned to train well who succeeds in a tournament but rather the one who is lucky and seizes it. Even the algorithms speak to us!

It is the end of this report **on Rock-Paper-Scissor using reinforcement learning**. We hope that you have appreciated and learned new thing in the field of machine learning. If you want to make your own skill on RL with this rps game, you can do this in your terminal as follows:

```
$ git clone https://github.com/batalong123/gym-rpsgame.git
$ cd gym-rpsgame
$ pip install -e .
```

Thank you for your reading!

@LB Tutorial College, Massock Batalong M.B.

Email: lbtutorialcollege@gmail.com

LinkedIn Network profile: Massock Batalong Maurice Blaise

Contact: lumierebatalong@gmail.com / maurice.batalong@aims-cameroon.org

Kaggle Data Science community: Massock Batalong Maurice Blaise – Notebook & Discussion Expert

Be free to learn don't worry!