


An ML Agent using the Policy Gradient Method to win a SoccerTwos Game

Victor Ulisses Pugliese ^a

Federal University of São Paulo, Avenida Cesare Mansueto Giulio Lattes, 1201, São José dos Campos, Brazil

Keywords: Reinforcement Learning, Proximal Policy Optimization, Curriculum Learning, Video Games.

Abstract: We conducted an investigative study of Policy Gradient methods using Curriculum Learning applied in Video Games, as professors at the Federal University of Goiás created a customized SoccerTwos environment to evaluate the Machine Learning agents of students in a Reinforcement Learning course. We employed the PPO and SAC as state-of-arts in on-policy and off-policy contexts, respectively. Also, the Curriculum could improve the performance based on it is easier to teach people in a complex gradual order than randomly. So, combining them, we propose our agents win more matches than their adversaries. We measured the results by minimum, maximum, mean rewards, and the mean length per episode in checkpoints. Finally, PPO achieved the best result with Curriculum Learning, modifying players' (position and rotation) and ball's (speed and position) settings in time intervals. Also, It used fewer training hours than other experiments.

1 INTRODUCTION

Artificial Intelligence (AI) plays an essential role in video games to generate responsive, adaptive, or intelligent behavior, mainly in non-player characters (NPCs), similar to human intelligence (Ranjitha et al., 2020). Thus, it keeps players engaged even when playing offline or when no players are available online.

Furthermore, several games provide interesting and complex problems for Machine Learning (ML) agents to solve, and gaming environments are secure, controllable, and offer unlimited valuable data for the algorithms. These characteristics make video games a perfect domain for AI research (Shao et al., 2019).

Therefore, the Artificial Intelligence Center of Excellence (Centro de Excelência de Inteligência Artificial - CEIA) professors at the Federal University of Goiás (Universidade Federal de Goiás - UFG) did a customized version of the SoccerTwos game and employed two ML baseline agents. The baseline agents were used to evaluate the students' agents in a Reinforcement Learning (RL) course.

The game simulates two soccer teams playing each other and counts who mark more goals in a specified time. Our goal was to identify which approach was the best recommendation to win the matches.


Thus, we proposed two Policy Gradient methods, Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC), because they are state-of-art in on-policy and off-policy ways, respectively. We also employed them with Curriculum Learning (CL). CL is a provocative learning strategy on how humans and animals learn better in a complex gradual order than randomly (Bengio et al., 2009).

To contextualize our work, we surveyed related works. Then, we performed an evaluation comparing the methods with the baseline agents of the CEIA/UFG. Finally, we present the main findings and conclude the paper.

2 BACKGROUND

2.1 Reinforcement Learning

Reinforcement learning (RL) is a subfield of machine learning (ML) that addresses the problem of the automatic learning of optimal decisions over time. It uses well-established supervised learning methods, such as deep neural networks for function approximation, stochastic gradient descent, and backpropagation, and applies it differently (Lapan, 2018) because there is no supervisor, only a reward signal, and feedback is delayed, not instantaneous. Therefore, an ML agent

^a  <https://orcid.org/0000-0001-8033-6679>

using these methods faces problems, learning its behavior through trial-and-error interactions with a dynamic environment (Kaelbling et al., 1996), communicate them through actions and states. Sutton and Barto (Sutton and Barto, 2018) model the reinforcement learning cycle as shown in Figure 1.

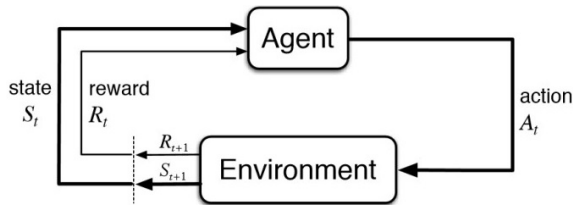


Figure 1: The agent–environment interaction in reinforcement learning (Sutton and Barto, 2018).

2.2 Policy Gradient Method

Policy Gradient methods are a reinforcement learning technique that optimizes parameterized policies concerning the expected return (long-term cumulative reward) per descending gradient (Huang et al., 2020). We selected two methods to work, and they are Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC)

2.2.1 PPO

Proximal Policy Optimization trains stochastic policy in an on-policy way, which means that it explores by sampling actions according to the latest version of its stochastic approach. We can implement this in either discrete or continuous action spaces (Sáenz Imbacuán, 2021) and (Achiam, 2018).

Furthermore, the method utilizes the actor-critic, which maps an observation to action, while the critic rewards that. So, It collects a set of trajectories for each epoch by sampling from the latest version of the stochastic policy. Then, It computes the rewards-to-go, and the advantage estimates to update the policy and fit the value function. The approach is updated via a stochastic gradient ascent optimizer, while the value function is via some gradient descent algorithm (Keras, 2022).

The amount of randomness in selecting actions depends on the initial conditions and the training procedure. The policy typically becomes progressively less random throughout training, as the updated rule encourages it to explore rewards it has already found (Sáenz Imbacuán, 2021).

2.2.2 SAC

Soft Actor-Critic optimizes stochastic policy in an off-policy way, forming a bridge between stochastic policy optimization and DDPG-style approaches. Initially, It was for environments with continuous action spaces, but there is already an alternative version for discrete ones (Achiam, 2018).

The method is based on the maximum entropy RL framework. Thus, The actor aims to maximize the expected reward while also maximizing entropy. In other words, It succeeds in the task by acting as randomly as possible. We can connect It to the exploration-exploitation trade-off: increasing entropy results in more exploration, accelerating learning later. (Achiam, 2018) and (Haarnoja et al., 2018).

Different from previous deep RL methods based on this framework formulated as Q-learning methods. SAC works like TD3, incorporating the clipped double-Q trick, but due to the inherent stochasticity of the policy in SAC, it also benefits from something like target policy smoothing. Therefore, It outperforms prior on-policy and off-policy methods in a continuous control benchmark (Achiam, 2018) and (Haarnoja et al., 2018).

2.3 Curriculum Learning

We implement those methods with a training strategy, such as *Curriculum Learning*. It is based on how humans and animals learn better in a complex gradual order than randomly (Bengio et al., 2009).

An easy way to demonstrate this strategy is to think about how math students learn arithmetic, algebra, and calculus in the education system. Teachers usually taught arithmetic before algebra, and algebra before calculus. The skills and knowledge learned in previous disciplines provide support for later lessons. We can also apply this principle in machine learning, where training the ML agents on the most straightforward tasks provides scaffolding for future challenging tasks (Camargo and Sáenz, 2021).

3 RELATED WORKS

We searched for the term 'SoccerTwos' on Google Scholar and found eight academic papers related to it. However, only six papers are about ML agents using Reinforcement Learning.

Sáenz wrote a master thesis about the impact of Curriculum Learning on the training process for an intelligent agent in a video game as the SoccerTwos case study, using the SAC and PPO algorithms.

To measure the performance, he used the mean cumulative reward. In some cases, this approach shortened the training process by 40% percent and achieved better measures than just algorithms. However, it was sometimes worse or did not affect other cases. PPO showed better results than SAC (Sáenz Imbacuán, 2021). Sáenz and Camargo published a paper in 2021 (Camargo and Sáenz, 2021), reporting a part of this thesis using PPO.

Majumder also realized a significant improvement in training when Curriculum Learning applied along with a Policy Gradient variant such as PPO. The incremental steps allow the agent to learn quickly in a new dynamic environment. Therefore, The authors recommended It in a competitive or collaborative context as SoccerTwos (Majumder, 2021).

Juliani et al. implemented a solution in a randomly generated multiagent using the PPO method in the ' Soccer Twos' environment. They trained the agents in a two-versus-two self-play mode. The agents learned to reposition themselves defensively or offensively and work cooperatively to score an opponent without conceding a goal (Juliani et al., 2018).

Osipov and Petrosian applied a modern multi-agent reinforcement learning algorithm using the TensorFlow library, explicitly created for SoccerTwos. They investigated different modeling tools and did computational experiments to find their best training hyperparameters. Furthermore, They applied this with the COMA gradient policy algorithm and showed Its effectiveness (Osipov and Petrosian,). Unfortunately, the authors wrote it in Russian, and we could not translate it.

Albuainain and Gatzoulis proposed an ML Agent, using reinforcement learning to adapt to dynamic physics-based environments in a 2D version of a vehicular football game. Thus, they perform behaviors such as defending their goal and attacking the ball using reward functions. They concluded that a reward function considering different state-space parameters could produce better-performing agents than those with less defined reward function and state-space (Albuainain and Gatzoulis, 2020).

4 EVALUATION OF THE METHODS USING ML AGENTS

We implemented our ML agents to play the SoccerTwos game customized by CEIA/UFG. The game is available at this GitHub

4.1 Explaining the Environment

The original SoccerTwos environment contains four players competing in a two vs. two toy soccer game, aiming to get the ball into the opponent's goal while preventing it from entering its own goal. The players have the same behavior parameters. The observation space consists of 336 corresponding to 11 ray-casts forward distributed over 120 degrees and 3 ray-casts backward distributed over 90 degrees each 6 possible object types, along with the object's distance. The forward ray-casts contribute 264 state dimensions and backward 72 state dimensions. The action space consists of 3 discrete branched actions (MultiDiscrete) corresponding to forward, backward, sideways movement, as well as rotation (27 discrete actions) (Tyagi, 2021).



Figure 2: Observation and action states of SoccerTwos Game.

The customized game has one time of 2 minutes, two ML agents that play each other (representing two teams), and a ball. Each team has two players as left and right. Both start within a pre-defined position, close to the field's middle as seen in Figure 3. Its reward function consists of two items (Oliveira, 2021):

- +1 - accumulated time penalty: when a ball enters the opponent's goal. With each fixed update, the accrued time penalty is incremented by (1 / MaxSteps). It reset to 0 at the beginning of an episode. In this build, MaxSteps is equal to 5000.
- -1: when ball enters team's goal.

4.2 The ML Agents Available by CEIA

In addition, the professors provided two baseline agents (CEIA DQN and CEIA PPO) to evaluate and



Figure 3: SoccerTwos Game by CEIA/UFG.

test the performance of student's experiments. Both agents do not use Curriculum Learning.

CEIA DQN is an ML agent that uses the Deep Q-Network method, which combines the neural network within a classical reinforcement learning method called Q-Network using the experience replay technique. They set optimized hyperparameters like 0.999 as eps decay, 336x512x27 as Q-Network, and 5000 as max steps.

CEIA PPO is an ML agent that uses the PPO method with these optimized hyperparameters like 256x256 as hidden layers and 5000 as rollout fragment length in a multiagent setting. The code is available at this GitHub. The ML agent is available at link.

4.3 Design of Experiments

Using Ray tools (v1.10.0) with Pytorch as a framework, we employed Policy Gradient methods with Curriculum Learning. Ray aims to provide a simple universal API for distributed computing, supporting multiple libraries to solve problems in machine learning, such as scalable hyperparameter tuning and industrial-grade reinforcement learning (Moritz et al., 2018).

Saenz recommended hyperparameter sets like [0.00001; 0.001] for learning rate, [128; 512] for batch size, [32;512] for hidden units and others for use in PPO and SAC methods (Sáenz Imbacuán, 2021). Also, there is an example of Ray example of Ray - link - applied in SoccerTwos, which uses 0.0003 for learning_rate, 0.95 for lambda, 0.99 for gamma, 256 for sgd_minibatch_size, 4000 for train_batch_size, 0.2 for clip_param, 20 for num_sgd_iter, two neural network layers to 512 units for PPO, and others.

We employed the experiments listed below:

- The first experiment only employs the policy method for 24 hours without the opponent's movement or Curriculum Learning. The method uses the recommended hyperparameters.

- The second experiment employs the policy methods with Curriculum A. It divides the 24 hours of training into 16 without the opponent's movement and 8 of a random opponent. The spontaneous activity happens in the middle of 16, making it three intervals of 8 hours. We also set new hyperparameters values.
- The last experiment employs the policy methods with Curriculum B. Thus, it sets different levels as Very Easy, Easy, Medium, and Hard, modifying players' (position and rotation) and ball's (speed and position) settings.

4.4 Evaluation Measures

To measure the performance of the Policy Gradient methods employed in this study, we use the metrics: *mean length per episode*; *mean*, *maximum*, and *minimum reward*.

- *Mean length per episode* refers to how many iterations the ML agent takes to complete a game move at a checkpoint.
- *Mean* is the average of cumulative reward values by checkpoints.
- *Maximum and minimum* are the biggest and lowest reward values by checkpoints.

5 RESULTS WITH THE ML AGENTS

This section presents the results of the Policy Gradient methods using Curriculum Learning for the SoccerTwos game.

We employed the 'PPO self-play' and 'PPO + Curriculum A'. The 'self-play' utilizes the Ray example hyperparameters, while we modified these settings for 'PPO + Curriculum A', removing the `train_batch_size`, `num_sgd_iter`, `rollout_fragment_length`, `no_done_at_end`, `evaluation_interval`, and `evaluation_num_episodes`, and we also updated the two neural network layers to 256 units. The Figure 4 shown the results comparing them performance.

As seen in Figure 4, both experiments learned to score, and their values are similar. However, if we observe the details, than the 'PPO + Curriculum A' (represented by orange, red, and blue colors) converges first. Also, It ended with a better mean reward than PPO self-play.

We also evaluate 'PPO + Curriculum A' versus the 'CEIA PPO', running the gaming 200 times. Our ex-

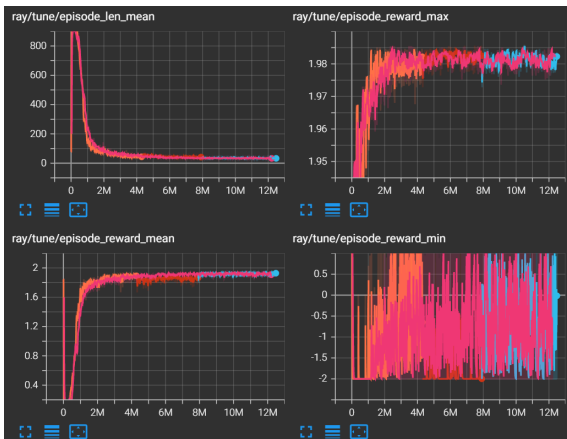


Figure 4: Results of PPO self-play (pink color) and PPO + Curriculum A (orange, red and blue colors).

periment had won 125 matches, which means 62.5%, of victories, as seen in Figure 5.

```

34
35   victor:
36     blue_team:
37       policy_blue_team_draws: 0
38       policy_blue_team_losses: 37
39       policy_blue_team_reward_max: 1.981600046157837
40       policy_blue_team_reward_mean: 0.47413596510887146
41       policy_blue_team_reward_min: -2.0
42       policy_blue_team_total_games: 100
43       policy_blue_team_win_rate: 0.63
44       policy_blue_team_wins: 63
45     orange_team:
46       policy_orange_team_draws: 0
47       policy_orange_team_losses: 38
48       policy_orange_team_reward_max: 1.9847999811172485
49       policy_orange_team_reward_mean: 0.42089200019836426
50       policy_orange_team_reward_min: -2.0
51       policy_orange_team_total_games: 100
52       policy_orange_team_win_rate: 0.62
53       policy_orange_team_wins: 62
54   policy_draws: 0
55   policy_losses: 75
56   policy_reward_max: 1.9847999811172485
57   policy_reward_mean: 0.44751399755477905
58   policy_reward_min: -2.0
59   policy_total_games: 200
60   policy_win_rate: 0.625
61   policy_wins: 125
    
```

Figure 5: Using CEIA PPO to evaluate the performance of PPO + Curriculum A.

We employed the 'PPO + Curriculum B' experiment, modifying players' (position and rotation) and ball's (speed and position) settings. We compare this one with 'PPO self-play', as seen in Figure 6.

As shown in Figure 6, the 'PPO + Curriculum B' convergence (blue color) is faster than PPO self-play (orange color). Thus, It achieved more than 1.8 by the mean reward of an episode in just 250k iterations, which did not happen with 'self-play'. Furthermore, the other measures are also better for it.

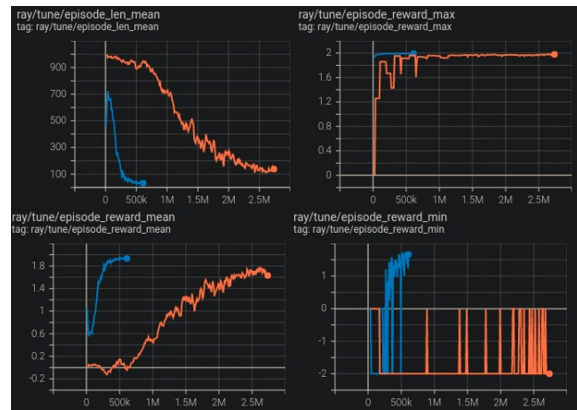


Figure 6: Comparing the performance of PPO + Curriculum B versus PPO self-play.

6 MAIN CONCLUDES

This study investigated Policy Gradient methods using the Curriculum Learning strategy, applied in a SoccerTwos game customized by CEIA/UFG. We employ PPO and SAC methods in this environment. Procedures were measured using minimum, maximum, average reward, and average episode duration metrics.

We had to deal with different challenges, such as the ML agent learning to move towards the ball, kick towards the opponent's goal to score a positive reward, defend our goal from the opponent, and others. Therefore, we recommend that an ML agent learns in a gradual order.

We obtained the best results in this game using the 'PPO + Curriculum B', executing its training in just 2 hours. We also found a better recommendation set of hyperparameters than Ray's example.

Unfortunately, despite the hyperparameters recommended by Saenz (Sáenz Imbacuán, 2021) for these methods, we did not achieve convergence for SAC experiments, as Ray's API returned an error message for some parameters like `buffer_init_steps`, `init_entcoef`, `save_replay_buffer`, `steps_per_update`. So, we did not show SAC results in this paper.

Next time, we will reproduce the Saenz (Sáenz Imbacuán, 2021) research using our hyperparameters recommendations as to future work. We also want to discover the recommended settings for SAC with Ray API for this game and continue evolving the Curriculum B strategy.

ACKNOWLEDGEMENTS

We would like to thank the CEIA/UFG professors for providing the game environment and support in the Reinforcement Learning course.

REFERENCES

- Achiam, J. (2018). Openai spinning up. *GitHub, GitHub repository*.
- Albuainain, A. R. and Gatzoulis, C. (2020). Reinforcement learning for physics-based competitive games. In *2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT)*, pages 1–6. IEEE.
- Bengio, Y., Louradour, J., Collobert, R., and Weston, J. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48.
- Camargo, J. E. and Sáenz, R. (2021). Evaluating the impact of curriculum learning on the training process for an intelligent agent in a video game. *Inteligencia Artificial*, 24(68):1–20.
- Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.
- Huang, R., Yu, T., Ding, Z., and Zhang, S. (2020). Policy gradient. In *Deep reinforcement learning*, pages 161–212. Springer.
- Juliani, A., Berges, V.-P., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., et al. (2018). Unity: A general platform for intelligent agents. *arXiv preprint arXiv:1809.02627*.
- Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.
- Keras, F. (2022). PPO proximal policy optimization.
- Lapan, M. (2018). *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd.
- Majumder, A. (2021). Competitive networks for ai agents. In *Deep Reinforcement Learning in Unity*, pages 449–511. Springer.
- Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., et al. (2018). Ray: A distributed framework for emerging {AI} applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 561–577.
- Oliveira, B. (2021). A pre-compiled soccer-twos reinforcement learning environment with multi-agent gym-compatible wrappers and human-friendly visualizers. <https://github.com/bryanoliveira/soccer-twos-env>.
- Osipov, A. and Petrosian, O. Application of the contract-structured gradient group learning algorithm for modeling conflict-controlled multi-agent systems.
- Ranjitha, M., Nathan, K., and Joseph, L. (2020). Artificial intelligence algorithms and techniques in the computation of player-adaptive games. In *Journal of Physics: Conference Series*, volume 1427, page 012006. IOP Publishing.
- Sáenz Imbacuán, R. (2021). Evaluating the impact of curriculum learning on the training process for an intelligent agent in a video game.
- Shao, K., Tang, Z., Zhu, Y., Li, N., and Zhao, D. (2019). A survey of deep reinforcement learning in video games. *arXiv preprint arXiv:1912.10944*.
- Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- Tyagi, D. (2021). Reinforcement-learning: Implementations of deep reinforcement learning algorithms and benchmarking with pytorch. <https://github.com/deepanshut041/reinforcement-learning>.