Project 2.1 - AI and Machine Learning

Group 12

Jan Ebenritter, Amir Mohseni, Allen Mirzoyan, Allan Nguyen, Aleksander Masojć, Noah Kestenholz, and Ron Adar

Maastricht University, Faculty of Science and Engineering

January 21, 2025

Abstract

This study explores how auditory inputs and restricted visual perception influence the performance of soccer agents trained in Unity's SoccerTwos environment. Additionally, it evaluates the effect of different learning rates on agent performance and assesses the reliability of Elo ratings as a performance metric by comparing self-play results with direct model-versus-model matches. The findings show that lower learning rates enhance performance when computational resources are sufficient, whereas higher learning rates are more effective under limited resources. While Elo ratings provide a useful heuristic for evaluating performance, they are not entirely predictive. These results highlight the importance of tailoring sensory inputs and training parameters to optimize agent performance in dynamic environments like SoccerTwos.

Contents

1 Introduction

This study explores reinforcement learning (RL) within the Unity ML-Agents framework, focusing on the SoccerTwos example where two teams of agents compete in a soccer environment. The investigation examines the impact of novel sensory configurations, particularly a sound sensor, against traditional visual sensors on agent performance. We assess how these sensors affect training efficiency and convergence rates and explore the influence of different learning rates on agent architectures.

Our primary question addresses whether integrating auditory information can improve agent strategy and performance more effectively than traditional methods alone. This question is relevant because it explores potential improvements in environmental awareness and decision making within multiagent systems.

The chosen algorithms, Proximal Policy Optimization (PPO) and Parameter Optimization with Conscious Allocation (POCA), are evaluated for their efficacy in this context, and POCA is is selected for its superiority in collaborative information-rich environments. The novelty of the study lies in testing the hypothesis that auditory signals can accelerate agent learning and adaptation.

This report is structured to first outline the experimental setup and methodologies used, followed by a detailed analysis of the experiments, results, and their broader implications for artificial intelligence and machine learning in complex and dynamic settings.

2 Methodology and Implementation

2.1 Environment

The environment used was the default SoccerTwos example scene in Unity [5]. In this environment, two teams, each with two agents, play against each other. The first stopping condition is when one team scores a goal. In this case, the scoring team will receive a positive reward, and the other team will receive a negative one. The second condition is when a maximum environment step count is reached, akin to the end time of a soccer match being reached with no goals scored. In this case, the game is a draw and the environment will reset.

2.2 Agents

The project focused on three types of agents.

- Default Agent: This agent is based on the original SoccerTwos environment soccer agent described in [5]. It features two ray-cast sensors, one pointing forward, and the other pointing backward.
- Reduced Agent: the same agent as the default without the backward pointing ray-cast sensor.
- New Agent: an agent with the forward pointing ray-cast sensor and a new sensor that simulates an ability to hear.

2.3 Sensors

2.3.1 Forward Ray-cast

The forward ray-cast sensor provides the agent with information based on rays that can detect objects. See Table 1. The forward ray-cast sensor works as shown in Figure 1.

Table 1: Forward ray-cast sensor parameters.

parameter	value
Rays Per Direction	5
Max Rat Degrees	60
Sphere Cast Radius	0.5
Stacked Raycast	3
Start Vertical Offset	0.5
End Vertical Offset	0.5
Detectable Tags	ball, purple goal, blue goal, wall, purple agent, blue agent

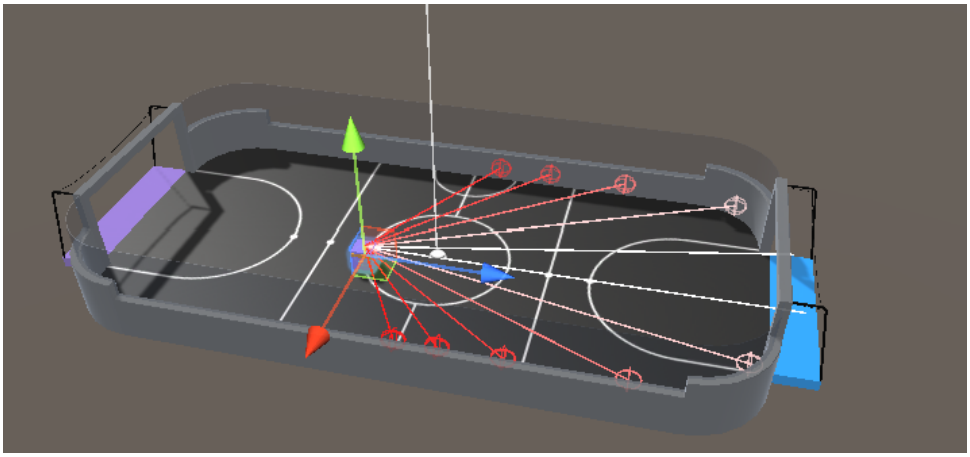


Figure 1: Illustration of the forward ray-cast sensor.

2.3.2 Backward Ray-cast

The backward ray-cast sensor parameters are shown in Table 2, and its visualization is shown in Figure 2.

Table 2: Backward ray-cast sensor parameters.

Parameter	Value
Rays Per Direction	1
Max Rat Degrees	45
Sphere Cast Radius	0.5
Stacked Raycast	3
Start Vertical Offset	0.5
End Vertical Offset	0.5
Detectable Tags	ball, purple goal, blue goal, wall, purple agent, blue agent

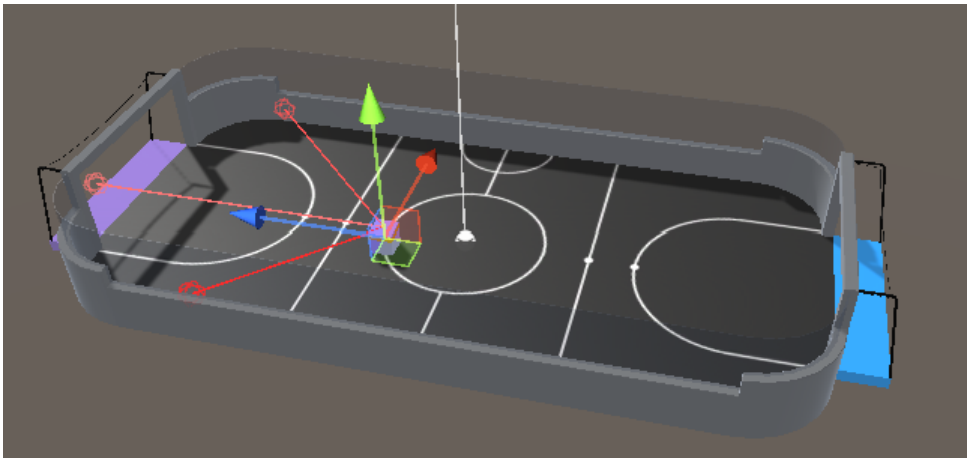


Figure 2: Illustration of the backward ray-cast sensor.

2.3.3 Sound Sensor

The sound sensor provides the agent with information about moving objects in a specific radius around it. The sensor provides data in the form of a Vector3 representing the relative location of moving objects inside the radius of the sensor. In addition, the sensor has "memory" of the past two observation frames. The memory should provide additional information to the agent and therefore give it a better idea of its surrounding environment.

Sound Sensor Implementation

The sensor works based on a list of all possible objects (agents and the ball). Each time the agent collects observations, the sensor will send it a list of all moving objects in range. Important points about the sensor are:

- Array size: due to Unity requirements the observation vector has to be the same size. If an object was not detected, the vector $(0, 0, 0)$ will be added instead.
- Observations memory: a queue containing the current and the latest two frames. Therefore, a total of three frames.
- The vectors: the information we choose to provide the agent with is the position of the detected object relative to the agent. To accomplish that the vector used is: $relativePosition = agent.position - detectedObject.position$
- The vectors: all vectors are normalized and the distance is added to the observation.

In Figure 3 you can see an illustration of the sound sensor.

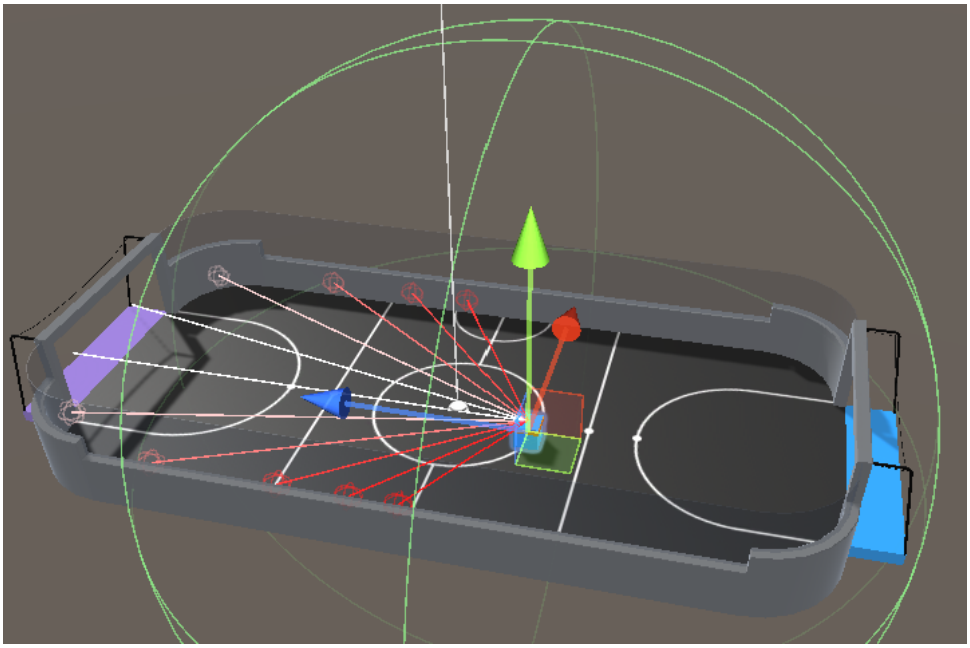


Figure 3: Illustration of the agent with the maximum hearing range.

2.4 Reinforcement Learning Algorithms

In our multi-agent environment, we evaluated two reinforcement learning algorithms. Proximal Policy Optimization (PPO) [6] is a well-established method recognized for its stability and reliability in training policies, showing great results in multi-agent environments [8]. However, Parameter Optimization with Conscious Allocation (POCA) [4] has been shown to outperform PPO in scenarios with significant information sharing among team players, such as soccer-like environments [1].

We chose POCA for our experiments, focusing on how different types of information, such as sound and visual cues, affect soccer agents and how these cues compare to each other in low—and high-compute training environments. We also conducted experiments with varying learning rates to explore their relationship with multi-agent training and determine whether different agent architectures exhibit significant differences under these conditions.

3 Experiments

3.1 Learning rates

For our first experiment, we train all three agent types using different learning rates. The goal of this experiment is to see how increasing the learning rate affects the agent's performance and the model's convergence based on its architecture. (The configurations are available in the Appendix A.4)

3.1.1 Setup

We compared the performance of the models using two different learning rates across our set of metrics. The learning rates were selected to evaluate performance across commonly used values and are as follows:

- $\alpha = 0.001$
- $\alpha = 0.0003$

Each model was trained to 20 million steps. The Unity environment was set up such that 32 soccer fields, and therefore games, would be used for training simultaneously as shown in Figure 4. The models were trained on two computers, one with an Nvidia RTX 3060 GPU and the other with an RTX 4060 GPU. We chose this because Nvidia GPUs are optimal for training agents due to their CUDA architecture's superior parallel processing capabilities, which enabled efficient simultaneous training of multiple agents. To ensure consistency in the comparisons, the learning rate was the only varying parameter across all experiments [2].

Figure 4: Training environment.

3.1.2 Metrics

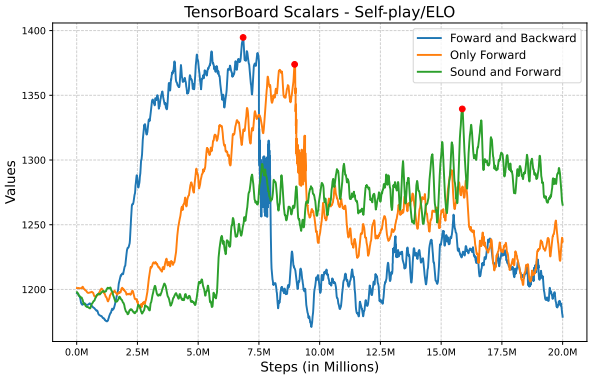
We tracked two main metrics to evaluate the models:

- **Episode length**: The duration until a team scores.
- **Elo rating**: A widely used rating system to measure the relative skill levels of players. Agents begin with a baseline Elo rating of 1200, which represents the average skill level. Elo scores are adjusted based on match outcomes during self-play, where agents play against their previous generations. A higher Elo rating indicates stronger performance [7].

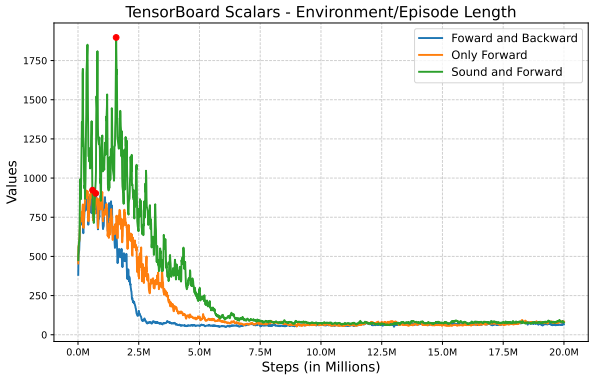
3.1.3 Results

Each model took roughly 17 hours to train (Appendix A.3). Therefore at least 102 hours of cumulative training time was required to run this experiment.

Learning Rate: $\alpha = 0.001$

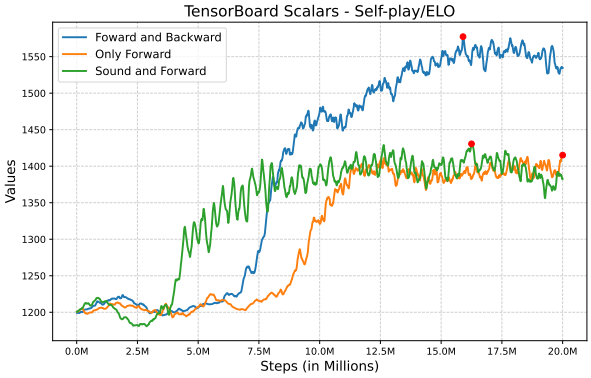


(a) Self-play Elo

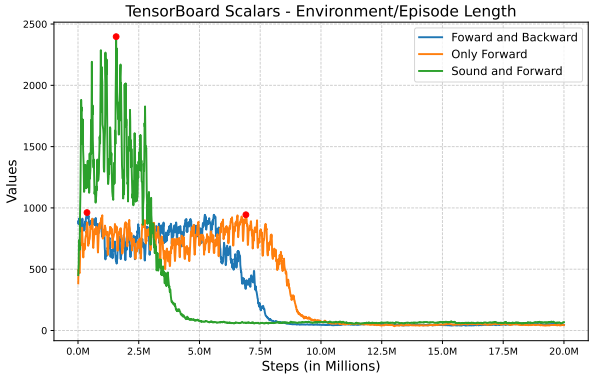


(b) Episode length

Learning Rate: $\alpha = 0.0003$



(c) Self-play Elo



(d) Episode length

Figure 5: Comparison of results for Self-play Elo and Episode length at different learning rates ($\alpha = 0.001$ and $\alpha = 0.0003$). The maximum amount reached on the y-axis is highlighted in red for each model.

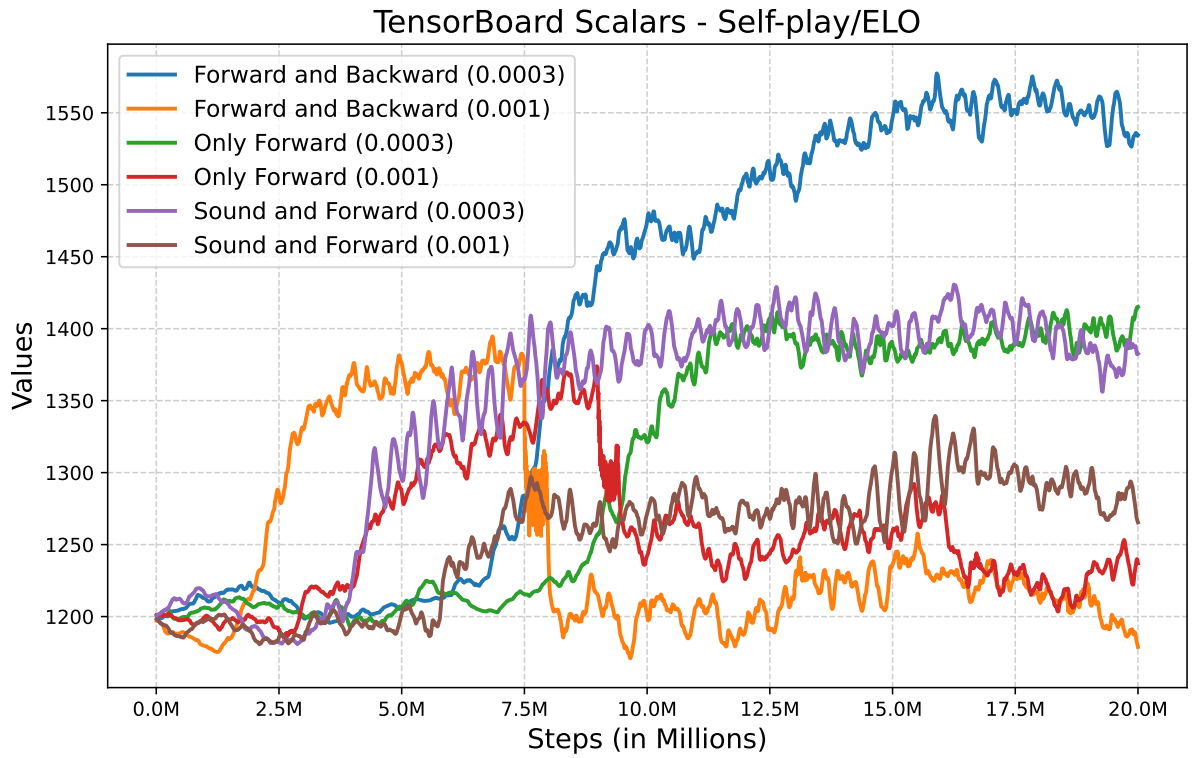


Figure 6: Comparison of Self-play Elo for all trained models.

3.1.4 Analysis

As we increased the learning rate, the models initially improved at a faster pace. However, their performance declined over time, suggesting that **higher learning rates hindered their ability to sustain improvements** in subsequent generations. Specifically, from step 7.5 million to 20 million, models trained with a high learning rate lost nearly all their earlier gains. In contrast, models trained with a smaller learning rate continued to improve gradually. This observation highlights a **trade-off between computational efficiency** (as lower learning rates require more training steps) and performance gains (Figure 6).[1]

Furthermore, our experiments used models with different input configurations. One model received rays from both forward and backward directions, another received only forward rays, and the third received forward rays combined with nearby sound data. We hypothesized that the **Forward and Backward model** would outperform the others due to the additional input data, and our results confirmed this for smaller learning rates, which aligns with findings reported in [3] (Figure 5c).

Interestingly, the **Forward and Sound model** outperformed the other configurations during the **first 7.5 million steps**. This suggests that incorporating sound data helped the model generalize faster and achieve rapid early improvements. However, as training progressed, the forward and backward model scaled more effectively, demonstrating yet another trade-off—this time between the type and amount of input data and the computational resources used for training (Figure 5c).

This trend is more evident in experiments with **lower learning rates**, as the models

[1]For additional plots, including the loss curves for the models, please refer to Appendix A.2.

trained with higher learning rates exhibited potential instability, which may have hindered their ability to generalize effectively. The instability could result from larger gradient updates leading to suboptimal adjustments, thereby masking the advantages of different input configurations. This observation is also reflected in the lengths of the episodes (Figures 5b and 5d).

3.2 Agent vs. Agent Simulations

In the second experiment, we evaluated the performance of the trained models by conducting matches between them. The goal was to test the hypothesis that models with higher Elo ratings perform better in soccer matches against models with lower Elo ratings.

3.2.1 Setup

We evaluated three types of models: Forward (F), Sound and Forward (S), and Forward and Backward (FB). Each type of model was trained using two different learning rates: 0.0003 (Group 1) and 0.001 (Group 2). These models were then matched against each other for a total of 120 games per pairing. The outcomes of the games were scored as follows: a win was recorded as $+1$, a loss as -1, and a draw as 0.

3.2.2 Metrics

To evaluate the models against each other, we measured the total scores of each model against all other models. Positive scores indicate a net advantage for the row model over the column model, while negative scores reflect the opposite. Diagonal cells (i, i) represent self-play scenarios, which were not evaluated and are shown as black cells in the table.

3.2.3 Results

Table 3 summarizes the results of the matches between the models. A legend is provided to clarify the abbreviations and groups.

Legend:

- **F1, S1, FB1:** Models trained with learning rate 0.0003.
- **F2, S2, FB2:** Models trained with learning rate 0.001.
- **F:** Forward-only model.
- **S:** Sound and Forward model.
- **FB:** Forward and Backward model.

	F1	**S1**	**FB1**	**F2**	**S2**	**FB2**
F1		16	-4	62	66	50
S1	-16		24	42	44	49
FB1	4	-24		55	74	68
F2	-62	-42	-55		-20	4
S2	-66	-44	-74	20		-20
FB2	-50	-49	-68	-4	20	

Table 3: Performance of models in head-to-head matches. Each cell represents the score of the row model against the column model over 120 matches. Diagonal cells (i, i) represent self-play scenarios and are shown as black cells.

Rank (Elo)	**Model (Elo)**	**Total Score Rank**	**Score (Row Sum)**
1	FB1	2	177
2	F1	1	190
3	S1	3	143
4	S2	6	-184
5	F2	5	-175
6	FB2	4	-151

Table 4: Comparison of model rankings based on Elo ratings through self-play (recorded at the 20 millionth step for all models based on Fig 6) and total scores from head-to-head matches. The table displays the rank, model, and scores for both metrics.

3.2.4 Analysis

The analysis reveals valuable insights into the relationship between Elo rankings derived from self-play and total scores obtained from head-to-head matches.

Firstly, models trained with a lower learning rate (0.0003) consistently outperformed their counterparts trained with a higher learning rate (0.001), as evidenced by their higher positions in both Elo rankings and total score rankings. This finding reinforces the results from our first experiment, suggesting that a lower learning rate may enhance stability and convergence during training, ultimately leading to superior performance when sufficient computational resources and training data are available.

Interestingly, the Elo rankings and total score rankings exhibit strong, though not perfect, alignment. For instance, FB1 ranked highest in the Elo ratings but second in total scores, whereas F1 achieved the highest total score but ranked second in Elo. This suggests that while Elo ratings serve as a reliable measure of self-play performance, they may not fully capture the models' competitive strength in head-to-head matches. These discrepancies highlight the complementary nature of these evaluation metrics and underscore the importance of using multiple perspectives to assess agent performance comprehensively.

4 Discussion

This study revealed key insights into the design and optimization of reinforcement learning systems in a simulated soccer environment with varied sensor configurations and learning rates. The initial increase in performance from integrating auditory sensors suggests that enhanced environmental perception can significantly impact agent performance. However, the long-term benefits of such enhancements did not persist, indicating potential drawbacks related to the complexity and computational demands of auditory data processing.

Lower learning rates facilitated more stable and gradual improvements, aligning with prior research that advocates for cautious hyperparameter tuning to avoid premature convergence to suboptimal policies. This finding underscores the delicate balance between learning speed and stability, which is crucial in settings where agents must adapt to dynamically changing environments.

The study also encountered limitations related to the computational overhead of advanced sensory processing, which negatively impacted the scalability of the findings. Future research should address these challenges by optimizing algorithms or exploring hardware solutions to fully leverage the potential of multimodal learning in complex environments. Similar challenges have been addressed in other modalities, as discussed in [3].

In essence, this research contributes to the understanding of how sensory data integration and learning parameter adjustments can enhance or impair the training of intelligent agents within the soccer tow environment.

5 Future Work

5.1 Reward Design

While the Elo rankings and score rankings show similarities (Table 4), the discrepancies between them suggest that the rewards used during training may not fully align with the development of optimal soccer-playing agents. Investigating the impact of reward design on agent performance and exploring alternative reward structures could provide valuable insights and serve as a promising area for future research.

5.2 Adaptive Learning and Sensory Modalities

Exploring adaptive learning rate schedules, incorporating additional sensory modalities, and experimenting with other reinforcement learning algorithms could provide deeper insights into training soccer agents under computational constraints. Such research underscores the value of sensory-driven approaches, careful parameter tuning, and robust evaluation in enhancing AI capabilities.

6 Conclusion

In this study, we utilized the Unity ML-Agents framework to train soccer agents in a simulated environment using the POCA algorithm. The primary focus was to examine how different sensory configurations and learning rates influence agent performance. We

studied the impact of varying learning rates and explored how different input and training data affect the optimal model and the computational resources required. Additionally, we assessed the utility of self-play Elo ratings as a metric for comparing model performance.

The results provide insights into the trade-offs involved in agent training and evaluation. Higher learning rates enabled faster initial performance gains but led to instability over extended training periods. Lower learning rates resulted in more gradual improvements, offering greater stability and better long-term performance. Furthermore, while self-play Elo ratings served as a useful heuristic for performance comparison, they were not entirely reliable as standalone metrics. These findings highlight the importance of adapting training configurations and evaluation methods to the specific goals and constraints of dynamic simulated environments.

References

[1] Andrew Cohen, Ervin Teng, Vincent-Pierre Berges, Ruo-Ping Dong, Hunter Henry, Marwan Mattar, Alexander Zook, and Sujoy Ganguly. On the use and misuse of absorbing states in multi-agent reinforcement learning, 2022.

[2] Dipesh Gyawali. Comparative analysis of cpu and gpu profiling for deep learning models, 2023.

[3] Viktória Ilosvay and Emanuele Iaccarino. *Unity ML Agents: Wall Jump and SoccerTwos environment using Reinforcement Learning (RL) tecnique*. PhD thesis, University of Deusto, 01 2024.

[4] Joshua Inman, Tanmay Khandait, Giulia Pedrielli, and Lalitha Sankar. Parameter optimization with conscious allocation (poca), 2023.

[5] Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. Unity: A general platform for intelligent agents, 2020.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[7] Unity Technologies. Elo rating system. `https://unity-technologies.github.io/ml-agents/ELO-Rating-System/`. Accessed: January 21, 2025.

[8] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022.

A Appendix

A.1 GitHub Repository

Our repository containing our work can be found at **GitHub**.

A.2 Learning Rate Experiment Results

This section provides additional details and plots related to the learning rate experiments conducted in this study. The results include visualizations such as the loss curves, performance metrics, and other supporting data. These plots offer more insight into the trends and observations discussed in the main text. The complete set of results can be accessed via the following **link**.

A.3 Training time

This section provides some log entries highlighting the training time elapsed until step 20 million was reached.

```
[INFO] SoccerTwos. Step: 20000000. Time Elapsed: 62022.649 s. Mean Reward: 0.044. Mean Group Reward: -0.011. Training. ELO: 1331.432.
[INFO] SoccerTwos. Step: 20000000. Time Elapsed: 60288.749 s. Mean Reward: 0.068. Mean Group Reward: 0.085. Training. ELO: 1383.869.
```

Figure 7: Example log entries

A.4 Default Training Configuration

This is the file we used to train our models. We only changed the learning rates in our experiments to ensure consistency.

```
default_settings: null
behaviors:
  SoccerTwos:
    trainer_type: poca
    hyperparameters:
      batch_size: 2048
      buffer_size: 20480
      learning_rate: 0.0003
      beta: 0.005
      epsilon: 0.2
      lambd: 0.95
      num_epoch: 3
      learning_rate_schedule: constant
      beta_schedule: constant
      epsilon_schedule: constant
    network_settings:
      normalize: false
      hidden_units: 512
      num_layers: 2
      vis_encode_type: simple
      memory: null
      goal_conditioning_type: hyper
      deterministic: false
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
        network_settings:
          normalize: false
          hidden_units: 128
```

```
          num_layers: 2
          vis_encode_type: simple
          memory: null
          goal_conditioning_type: hyper
          deterministic: false
    init_path: null
    keep_checkpoints: 5
    checkpoint_interval: 500000
    max_steps: 50000000
    time_horizon: 1000
    summary_freq: 10000
    threaded: false
    self_play:
      save_steps: 50000
      team_change: 200000
      swap_steps: 2000
      window: 10
      play_against_latest_model_ratio: 0.5
      initial_elo: 1200.0
    behavioral_cloning: null
env_settings:
  env_path: null
  env_args: null
  base_port: 5005
  num_envs: 1
  num_areas: 1
  seed: 1234
  max_lifetime_restarts: 10
  restarts_rate_limit_n: 1
  restarts_rate_limit_period_s: 60
engine_settings:
  width: 84
  height: 84
  quality_level: 5
  time_scale: 20
  target_frame_rate: -1
  capture_frame_rate: 60
  no_graphics: false
environment_parameters: null
torch_settings:
  device: null
debug: false
```

Listing 1: Default Training Configuration