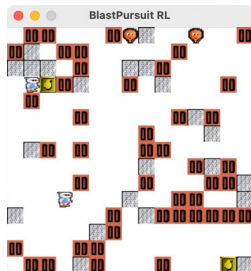


BlastPursuit

Multi-Agent Systems Project

Guido Laudenzi (0001033343), Sami Osman (0000929408)



- ① Introduction
- ② Game Implementation
- ③ Algorithm Implementation
- ④ Conclusions

- 1 Introduction
- 2 Game Implementation
- 3 Algorithm Implementation
- 4 Conclusions

Abstract

- The project deals with the integration of classic Bomberman game elements with advanced multi-agent system concepts.
- Implementation of **Reinforcement Learning** (RL) algorithms for intelligent decision-making by agents.
- Wants to provide an engaging gaming experience while also serving as an educational tool for AI applications.
- Demonstration of emerging behaviors through reinforcement learning in dynamic, multi-agent settings.

Introduction

- The project aim to exploit the synergy of classic Bomberman gameplay with AI-driven **multi-agent systems**.
- Emergence of strategic behaviors through reinforcement learning in a grid-based environment.
- Developed using *Python*, *Pygame*, and OpenAI's *gym* [1] environment for robust game simulation and AI training.
- Implementation of advanced algorithms like *PPO* [2], *SAC* [3], and *DQN* [4] from the *Stable-Baselines3* [5] library to enhance AI learning and gameplay experience.

Background and Motivation

- **Original Bomberman game:** A classic with simple yet engaging gameplay, players navigate maze-like grids, strategically placing bombs to destroy walls and opponents while avoiding being caught in explosions.
- **Augmented version with Multi-Agent Systems:** Integration of traditional gaming with modern AI technology, introducing strategic complexity through AI-driven decision-making and learning.
- **BlastPursuit:** A platform for studying AI and multi-agent systems, populated with non-standard intelligent agents interacting and learning through RL.

Reinforcement Learning (RL)

- RL in *BlastPursuit* was employed to train agents for intelligent navigation within the game environment.
- **Q-learning** is a fundamental RL algorithm used to estimate action quality and update Q-values iteratively based on cumulative rewards [6]:

$$Q(s, a) \leftarrow (1 - \alpha) \cdot Q(s, a) + \alpha \cdot \left(r + \gamma \cdot \max_{a'} Q(s', a') \right)$$

- Cooperative and competitive scenarios introduce complexity, requiring agents to adapt to different strategies.
- RL demonstrated effectiveness in various fields such as videogames [7], autonomous driving [8] or robot control [9], highlighting its potential for addressing real-world challenges.

Related Work

- We aim to explore multi-agent deep reinforcement learning (MARL) in game environments, with a focus on competitive settings.
- Related papers focus on the same kind of agents:
 - ① The agents in *Pommerman* [10] must learn to collaborate towards shared goals while navigating around the grid-world.
 - ② [11] introduces the notion of Long Short-Term Memory (LSTM) and various state representations for training a PPO algorithm.
- Here, one model controls agents from two distinct teams with different goals, diverging from traditional single-agent setups.

- ① Introduction
- ② Game Implementation
- ③ Algorithm Implementation
- ④ Conclusions

Game Description

- **Bombers** navigate a grid with indestructible and destructible walls, strategically placing bombs to clear paths and eliminate Killer agents.
- **Killermen**, the opposing agents, aim to survive and disrupt Bombers' strategies, adding unpredictability and challenge to the game.
- Bombers must avoid explosions and evade Killermen to survive, highlighting the tactical interplay between the two types of agents.
- Game concludes when all Killermen or all Bombermen are eliminated.

Grid-world Representation

Agents –Bombers and Killers:

- **Bombers** aim to clear paths, target Killers, and place bombs.
- **Killers** aim to survive and chase and catch Bombermen.

Grid environment with different cell types:

- **Empty Cells** provide freedom of movement for all agents, crucial for navigation.
- **Indestructible Walls** shape gameplay, forcing agents to find alternate routes and adapt to constraints.
- **Destructible Walls** can be destroyed by bomb explosions, altering the grid layout, opening new pathways.
- **Bombs** have fixed countdown and blast radius, used for clearing destructible walls and eliminating Killers (posing a threat to their owners!).

- 1 Introduction
- 2 Game Implementation
- 3 Algorithm Implementation**
- 4 Conclusions

Technical Details

- Implemented 7×7 and 11×11 grid environments for evaluation of different RL algorithms: *PPO*, *SAC* and *DQN*.
- *Gymnasium* [12] toolkit facilitates shaping of AI agents, allowing easy representation and debugging of key agent characteristics such as state, actions, and rewards.
- *PettingZoo* [13] framework for extension to multi-agent environments; *SuperSuit* [14] to interface with *Stable-Baselines3* algorithms.
- *Pygame* for dynamic and accurate visual representation of the game state, ensuring engaging interface for players and experimenters.

RL Setup 1/3

- **Algorithmic Settings:** Standard hyperparameters used for RL algorithm comparison, with agents trained for 15M steps and evaluation episodes occurring every $15,000 \times N$, with N number of agents.
- **Observation State:** Each agent's observation O_i comprises a matrix representing elements placed in the environment as first layer and the agent's position in the second, enriched with information from the previous 3 time-steps (overall shape of $S \cdot 4 \times S \times 2$ where S grid size).

RL Setup 2/3

- **Actions:** Agents have the freedom to move in four directions and Bomberman possess the additional ability to place bombs ($a_i \in [0, 1, 2, 3, 4]$).
- **Rewards:** Crafted reward functions to incentivize desired behaviors, ensuring fair competition and parameter sharing among agents, with considerations for various scenarios including time-out, kills, wall destruction, bad moves, deaths, wins, exploration, bomb efficacy, and bomb radius proximity.

$$R_B = R_t + R_k + R_w + R_b + R_d + R_{win} + R_e + R_n + R_r$$

$$R_K = R_t + R_k + R_b + R_d + R_{win} + R_r$$

RL Setup 3/3

- **Neural Approximator** has two 2D Convolutional Layers with ReLU activation, mapping input images to 8 and 16 feature maps. A linear layer condenses these into a 64-length vector.
- In PPO and A2C, the actor-critic network is shared, while in DQN, actions are directly outputted.

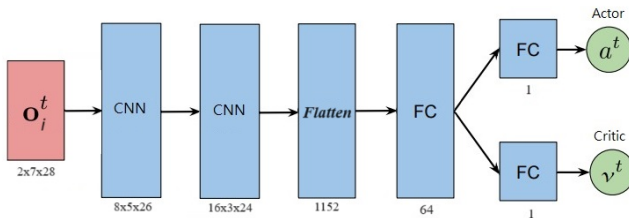


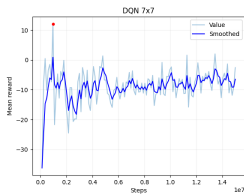
Figure 1: Actor-Critic Neural approximator in case of 7×7 observation

- 1 Introduction
- 2 Game Implementation
- 3 Algorithm Implementation
- 4 Conclusions

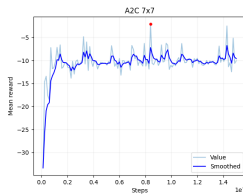
Results

- Teams of 3 agents were employed across different grid-world sizes and algorithms, ensuring comprehensive evaluation of model performance.
- The multi-agent DRL model demonstrated **promising results** during training, showcasing effective learning of environment dynamics, with Bomberman's inclination at placing bombs to eliminate opponents.
- Unexpected behaviors emerged, with Bomberman displaying suicidal tendencies impacting overall model performance.
- A cause could be the potential **misalignment** in the reward function: it is a challenge to accurately shape the reward function for competitive objectives.

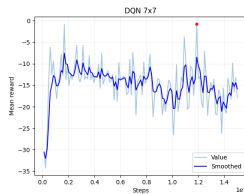
Convergence



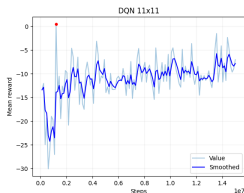
(a) DQN, 7×7 grid



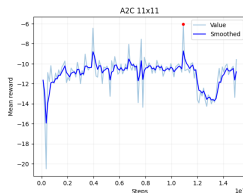
(b) A2C, 7×7 grid



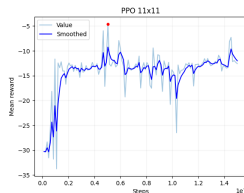
(c) PPO, 7×7 grid



(d) DQN, 11×11 grid



(e) A2C, 11×11 grid



(f) PPO, 11×11 grid

Conclusions

- We have successfully developed a multi-agent DRL model for *BlastPursuit*, a variant of Nintendo's Bomberman game, including various agent types with distinct objectives.
- Model exhibited promising **convergence outcomes**, with agents learning how to maximize the reward function within the multi-agent setting.
- Challenges arose during the evaluation phase, highlighting the need for further refinement.
- Future improvements could include refining the reward function to align with desired behaviors while exploring alternative training strategies like **adversarial training**, in which two distinct models learn how to control each team.

Additional Content

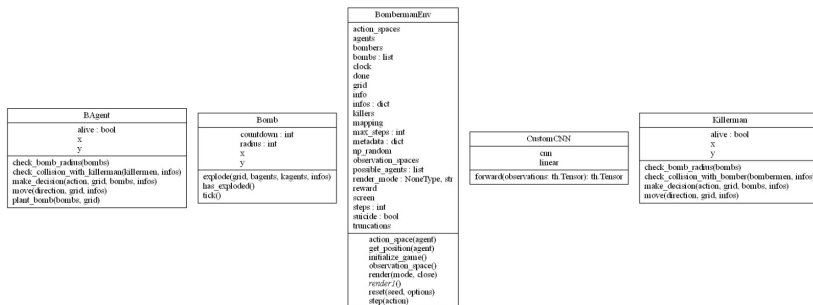


Figure 3: UML Entity diagram generated with **Pyreverse**

The End

Thank you for your attention.

References I

- [1] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.
- [2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
- [3] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*, pp. 1861–1870, PMLR, 2018.
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.
- [5] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.
- [6] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. A Bradford book, MIT Press, 1998.
- [7] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.
- [8] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.
- [9] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 6023–6029, IEEE, 2019.
- [10] C. Resnick, W. Eldridge, D. Ha, D. Britz, J. Foerster, J. Togelius, K. Cho, and J. Bruna, "Pommerman: A multi-agent playground," *arXiv preprint arXiv:1809.07124*, 2018.

References II

- [11] . Goulart Faria Motta França, A. Paes, and E. Clua, *Learning How to Play Bomberman with Deep Reinforcement and Imitation Learning*, pp. 121–133. 11 2019.
- [12] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. T. J. Shen, and O. G. Younis, "Gymnasium," Mar. 2023.
- [13] J. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. S. Santos, C. Dieffendahl, C. Horsch, R. Perez-Vicente, *et al.*, "Pettingzoo: Gym for multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 15032–15043, 2021.
- [14] J. K. Terry, B. Black, and A. Hari, "Supersuit: Simple microwrappers for reinforcement learning environments," *arXiv preprint arXiv:2008.08932*, 2020.