Adaptive Agents in 1v1 Snake Game: Exploring Dynamic Environments

1st Hampus Fink Gärdström
MMMI TEK
University of Southern Denmark
Odense, Denmark
hgard20@student.sdu.dk

2nd Henrik Schwarz
MMMI TEK
University of Southern Denmark
Odense, Denmark
hschw17@student.sdu.dk

***Abstract*—This paper investigates the adaptability of PPO-trained agents in a dynamic environment. Usually an agent is trained in a specific environment and adapts to it, learning to get better and how to navigate it to perform better. Changes in this environment may result in deficiencies in the performance of these trained agents. Current research does not detail how the training of agents impact the adaptability of agents in different environments and what parameters impact this. It is pivotal to investigate this to create more versatile intelligent agents.**

The aim of this study is to explore this area and investigate how training agents on different environments will affect the adaptability of agents when introduced to new environments it has not been trained on.

To achieve this 36 models were trained using 36 configurations to play one-versus-one(1v1) Snake and later compared against each configuration to measure the adaptability of the each model.

The results show that some distinct parameters substantially affect the adaptability of agents in differing environments. Some parameters were not important and did not affect the training nor adaptability of agents in different environments.

Furthermore the results showed that the most adaptive agents were the ones not trained on the most expansive and complex environment, but the simplest. Investigating which parameters have an impact and how to maximize the adaptability of an agent is crucial in the development of adaptive agents, the results of this study provides and insight into this and in the context of 1v1 Snake.

***Index Terms*—snake game, multi-agent, reinforcement learning, adaptive agents, proximal policy optimization**

I. Introduction

Reinforcement learning with multi agent systems (MAS) in games is an interesting area of research that focuses on how to train agents using self-play and using only the state of the game and possible actions. There has been impressive results from companies such as Google DeepMind with their AlphaGO Zero algorithm [1] that mastered the game of Go within 24 hours and beat the Starcraft 2 world champion [2]. Algorithms such as Q-learning and Proximal Policy Optimization(PPO) have become popular in recent years. There is research that examines the performance of reinforcement trained agents in static environments and using Q learning algorithms on the other hand in this paper the focus is on using the PPO learning algorithm [3] to investigate how well the trained model can adapt to different environements where the game changes. The context of this is done is using the game of 1v1 Snake. The 1v1 Snake game is inspired by the international BattleSnake [4] where different AI models fight each other in a game of multiplayer snake, which has the same rules as classic snake. The added factor of multiple snakes introduces interesting complexity, however to keep the scope of the study limited it was constrained to one versus one.

The aim of the study is to explore the impacts of training agents in a specific environment, and comparing how well the agent fares in a different environment, than the one it was trained on. By conducting a comparative analysis between the agents proficiency and their ability to adapt to different environments we aim to discern patterns, challenges, and potential limitations associated with the adaptability of the trained agents.

Our research endeavors to contribute to the broader understanding of agent adaptability and transfer learning. This knowledge has the potential to support the development of more robust and versatile intelligent agents, capable of transitioning between differing environments and task, ultimately enhancing their applicability and use.

To this end, this paper aims to address the following research questions:

RQ1: How do variations in game parameters, such as map size, maximum amount of food at once, the presence of walls, and the quantity of walls, impact the performance of Proximal Policy Optimization (PPO)-trained agents in a 1v1 Snake game?

RQ2: How can the performance and adaptability of an agent be quantitatively compared when deployed in different scenarios?

RQ3: To what extent does changes in the game parameters influence the adaptability of a trained agent in different scenarios?

The remainder of the paper is structured as follows. Section II will describe related work and cover existing research and challenges on reinforcement learning for the game Snake, and games in general. Section III will describe the research method. It covers the implementation of the snake game, feature design, reward structure, game settings, model training and the construction of the experiment. In section IV the results from the experimented are detailed. The discussion in section V will analyze the findings and aim to answer the

research questions. Finally the conclusion and future work will be covered in section VI.

II. Related work

In this section the concept of reinforcement learning and relevant algorithms are briefly introduced. Additionally, an overview of existing research on the Snake game is presented for context and reference.

Reinforcement learning is paradigm of the field of machine learning where the learner is not told which actions to take but has to discover by it on their own by using the state of the problem and selecting actions based on what yields the highest reward. To gather reward a reinforcement learning agent has to try actions it has done in the past but also explore new options to also make better actions in the future. [5, p.1-13].

Previous studies [6] have applied reinforcement learning to games of snake, achieving a highscore of 66 and covering 46% of the map using deep-neural network approach called Deep Q Network(DQN) with a focus on adjusting learning parameters. These parameters were time spent training, the neural network configuration and memory of the agent. To performance of agents were evaluated by generating CSV files, which recorded the highscore and median score across different training configurations.

Another paper [7] used Q learning to create a self playing agent for snake where three different reward mechanisms were used. The first reward mechanisms involved the distance to the nearest fruit, where the agent received rewards or penalties based on the proximity to the next fruit for each iteration. The second mechanism was discouraging idling by penalising the agent if the snake did not eat a fruit within the a certain amount of steps. The third was a training gap for the nodes in the network. They took the three reward mechanisms and aggregated them as one value. To evaluate the performance of the agent the they played a 1.000 games and averaged the score and the length of steps taken for later analysis.

This paper [8] compares the performance of using PPO and DQN learning algorithms using the game of snake. Initially they tried both algorithms and ended up selecting the DQN algorithm based on PPO random sampling performance not being adequate. Their model optimization included five parameters: visibility of the snake body in that direction, visibility of food in that direction, distance to its body, distance to food and distance to the wall. The sampling of these five parameters were initially the eight adjacent squares around the snakes head, which later was deemed insufficient leading the authors to expand the sampling to be in 16 directions, being two boxes to left, right, up, down and all four diagonal directions around the snakes head.

III. Method

In this section the implementation of the game, the feature design and the reward structure of the agent is detailed. Additionally, the details on how and which models were trained are described along with how the adaptability experiment was performed.

A. Snake implementation

The game was implemented as a multi-agent PettingZoo [9] environment for 2 agents. In algorithm 1 pseudo-code for the step function of the game is shown. The function is run at each update of the game. It takes the actions of each snake as input, and for each snake derives the next position based on their respective action. Depending on whether certain conditions are met the snake is rewarded or terminated. If the snake collides with an obstacle or goes out of the map bounds it is terminated and negatively rewarded. If it discovers food it is rewarded, if not it is instead negatively rewarded and loses it tail. Depending on whether the snake got closer to food it is also rewarded.

Finally, the next position is set as the new head of the snake. Once each snake is processed, walls and food is added based on a 20% chance, and given that the max limit of each item has not been reached inside the map. The placement is random. In case the max limit of walls has been reached then a random wall is removed and a new wall is placed in a random location.

Terminated agents do not receive further actions and are therefore no longer updated. The game lasts until both agents have been terminated or a set number of max game iterations has been reached. In order to limit the runtime of the game and prevent the model running each episode too long a limit of 5.000 step iterations was set for each game run, and if it was exceeded the game will terminate all agents thereby ending the game.

B. Features Design

The feature design was inspired by the a study [8] mentioned in the background section. The state input for the learning algorithm is the same where the 16 directions were chosen and the same five observation parameters were chosen due to their demonstrated adequacy, however they were adjusted slightly after testing to perform better in this implementation of snake.

The first two parameters visibility of the snake body and food are boolean values (1 and 0) that describes whether a part of the snakes body or a piece of fruit can be found in the square. This provides the algorithm with immediate information of the surroundings of the snake.

The next three parameters provide the snake with a representation of the global state of the game without having to provide the whole board. The distance to body uses the Manhattan Distance[1] to calculate the distance to the nearest body part and should help the algorithm with understanding confinement and optimizing for having available state. Similarly distance to the nearest food and distance to the nearest wall, are both also defined using Manhattan distance. These parameters are designed to optimize finding food and avoiding collision with walls.

The action space for the game is defined as an integer value from 0 to 3 that are defined as up(0), down(1), left(2) and right(3).

[1] Manhattan Distance Wikipedia

Algorithm 1 Step Function

function STEP(actions)
  if not actions **then**
    return $\{\}, \{\}, \{\}$
  end if
  $rewards \leftarrow \{\text{agent} \mid \text{agent} \in \text{agents} : 0\}$
  $observations \leftarrow \{\text{agent} \mid \text{agent} \in \text{agents} : \text{getObservation(agent)}\}$
  $terminations \leftarrow \{\text{agent} \mid \text{agent} \in \text{agents} : \text{False}\}$
  for each agent $\in$ agents **do**
    $snake\_head \leftarrow$ getHead(agent)
    $next\_pos \leftarrow$ move($snake\_head, actions$[agent])
    if posInvalidOrBlocked(next_pos) **then**
      killAgent(agent)
      $rewards$[agent] $\leftarrow -29$
      $terminations$[agent] $\leftarrow$ True
      continue
    end if
    if posIsFood(next_pos) **then**
      $rewards$[agent] $\leftarrow 27$
    else
      removeTail(agent)
      $rewards$[agent] $\leftarrow -0.3$
    end if
    if agentGotCloserToFood(next_pos, agent) **then**
      $rewards$[agent] $\leftarrow 0.3$
    end if
    addHead(next_pos, agent)
  end for
  addFoodRandomly()
  addWallsRandomly()
  return $rewards, observations, terminations$
end function

C. Reward Structure

Rewards were on based on conditions described by Yuhan [8] where the initial values were the same and subsequently adjusted based on preliminary testing. The preliminary testing adjustments resulted in setting the reward for obtaining food at 27, a penalty of -0.3 for in-action i.e. not eating food, -29 for a death and 0.3 reward for moving closer to food.

D. Model training

A total of 36 game scenarios, or game parameter combinations, were devised from a range of parameters listed in table I, where all permutations that change the environment were included. The game parameters were selected as they were envisioned to significantly impact the behavior and need of the agent and were therefore suitable for the experiment on adaptability. The implementation of PPO from Stable Baselines3 [10] using the MlpPolicy was used to train the models. The hyperparameters underwent brief testing and changes were made to the step size and batch size to speed up training time, the used hyperparamters are listed in table II; the entries in bold are the ones that differ from the default configuration. All models were trained to 25 million total timesteps using a CPU device. The total amount of timesteps was selected, as during preliminary testing, it appeared to provide adequate results and it made it possible to train all 36 models within a reasonable time frame.

TABLE I
GAME SETTINGS

Parameter	Variables
Map Size	5, 11, 19
Food Total Max	2, 10, 15
Walls Enabled	False, True
Walls Max	2, 10, 15

TABLE II
HYPERPARAMETERS

Hyperparameter	Value	Hyperparameter	Value
Policy	***'MlpPolicy'***	***Batch Size***	***8000***
Learning Rate	0.0003	GAE Lambda	0.95
n_steps	***32000***	Clip Range	0.2
n_epochs	10	Gamma	0.99
Normalize Advantage	True	Entropy Coefficient	0.0
VF Coefficient	0.5	Max Grad Norm	0.5
Use SDE	False	SDE Sample Freq	-1
Rollout Buffer Class	None	Rollout Buffer Kwargs	None
Target KL	None	Stats Window Size	100
Tensorboard Log	None	Policy Kwargs	None
Verbose	0	Seed	None
Device	**'cpu'**	_init_setup_model	True

E. Experiment

The experiment was executed by running each model against every combination of game parameter configurations listed in table III. This totalled 1296 (36 models $\times$ 36 configurations) different model combination pairings, and each pairing was evaluated for 500 episodes running through an entire game, from which average and max metrics were collected. These values were collected into a CSV-file for processing.

The metrics collected were the average of the total reward, snake size, food eaten and moves taken for every agent across all. Moreover, the maximum of these values across all episodes of an evaluated model $\times$ configuration pairing was also recorded.

IV. RESULTS

The results of the experiment produced multiple graphs for analysis.

An array of bar charts showcasing the average performance of each model across all game configurations has been made. Here, every metric for each model has been averaged over every game parameter configuration it has been evaluated against. The bar charts can be seen in fig. 1. The data shows that certain models perform consistently worse or better than others across all game parameter configurations, specifically models trained on combinations with larger maps sizes compared to smaller. Models 9 to 13, 21 to 24 and 33 to 36, all

TABLE III
PARAMETER CONFIGURATIONS

Id	Food Max	Map Size	Walls Enabled	Max Walls
1	2	5	False	0
2	2	5	True	2
3	2	5	True	10
4	2	5	True	15
5	2	11	False	0
6	2	11	True	2
7	2	11	True	10
8	2	11	True	15
9	2	19	False	0
10	2	19	True	2
11	2	19	True	10
12	2	19	True	15
13	10	5	False	0
14	10	5	True	2
15	10	5	True	10
16	10	5	True	15
17	10	11	False	0
18	10	11	True	2
19	10	11	True	10
20	10	11	True	15
21	10	19	False	0
22	10	19	True	2
23	10	19	True	10
24	10	19	True	15
25	15	5	False	0
26	15	5	True	2
27	15	5	True	10
28	15	5	True	15
29	15	11	False	0
30	15	11	True	2
31	15	11	True	10
32	15	11	True	15
33	15	19	False	0
34	15	19	True	2
35	15	19	True	10
36	15	19	True	15

seem to perform under the other models that have a smaller map size. Model 11 with a max wall size of 10 also performs consistently better better than model 10 and 12 with max walls parameter 5 and 15. When comparing other models that also have a max wall parameter of 10 and their neighboring models with parameters of 5 and 10 the same cannot be found.

The average value charts seem to be more consistent with less variance where the with max value have a bigger variance in value.

Several heat maps were created correlating trained models and game parameter configuration to see the performance of each model and configuration pairing on some metric. The heatmaps included in these result showcase the average moves and average total reward.

The average moves correlation matrix shown in fig. 2. The matrix shows distinct patterns of game parameter combinations that under perform comparatively. In particular, the game parameter combinations from the series 1 to 4, 13 to 16 and 25 to 28 exhibited a very low amount of moves for all trained models. All of these game parameters settings had a commonality as they all had a map size of 5 $\times$ 5. Some combinations, however, resulted in a high rate of moves being taken. These combinations were 10 to 12, 17, 21 to 24, 33 to 36, this includes all combinations where the map size is 19 $\times$ 19 with the exception of combination 9 and the addition of combination 17.

The models that resulted in a substantially lower amount of moves were the models from 10 to 12, 21 to 24 and 33 to 36. All these models had a commonality in that they were trained on a map size of 19 $\times$ 19. However, model 9 that was also trained on the large map size did not exhibit a similarly poor performance. Model 17 did however show a similar lack of moves taken, in a way compared to the other models that under performed. No specific models showed any comparatively higher performance.

The average total reward correlation matrix is shown in fig. 3. The average total reward is the total cumulative reward form the reward structure an agent achieves over its lifetime. These values were then averaged across all games played in the model $\times$ game parameter configuration pairing, like was done with the other metrics shown in the previous heatmap and the averaged values in the bar charts. Similar to the other heatmap, the data shows that game parameter configurations with a map size of 19 $\times$ 19 as well as the models trained on them result in a poor average total reward compared to all others.

Moreover, the total reward for the different trained models, with the exception of the aforementioned under performing ones, all exhibit a similar average total reward whereby the resulting value appears mostly dependent on the game parameter configuration. The game parameters configurations that result in noticeably good scores are the configurations from 1 to 4, 13 to 16 and 25 to 28. A shared commonality between all of these combinations is that the map size is set to 5 $\times$ 5.

V. DISCUSSION

In this section the results are analyzed with the aim of answering the study's research questions.

The results from the heatmaps suggest that the map size was the primary factor in affecting the performance of the trained agent. It was shown that models trained in a configuration with a large map size performed poorly across all other game parameter configurations. It can be reasoned that the cause of this may be that the time to for the agent to learn how to navigate within a large environment is more expansive, and as such it performs poorly given the amount of time it has been trained, as such it might be that increasing the training time would result in a better performance. Nonetheless, a large map size negatively improves the performance.

The game parameters of food max and max walls did not show any substantial impact on the training for the model nor the difficulty for other models to adapt to the environment. This is interesting as it was expected that having much more food on the map would provide with a significantly higher score, and that having many walls would make it much harder for the agent to perform well, in particular in maps where the size was small. However, the results suggest that this was not

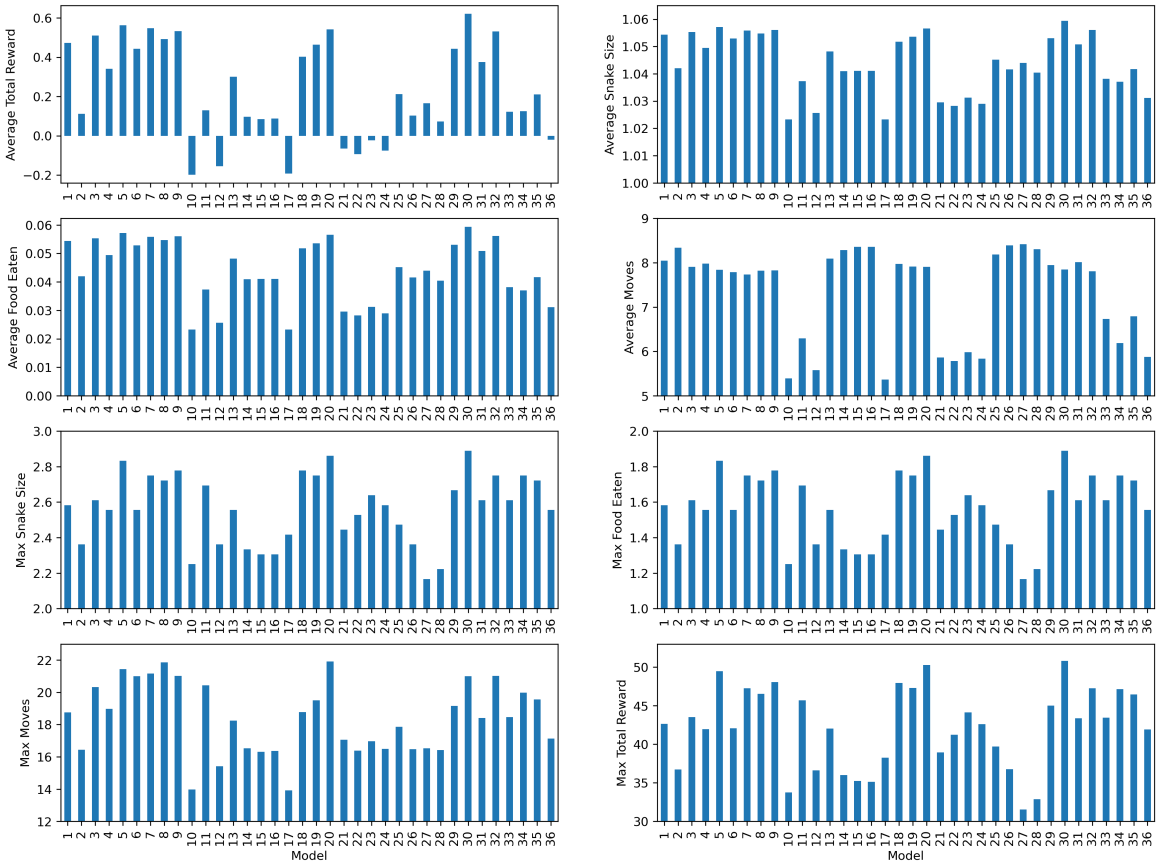

Fig. 1. Bar charts of all the measured metrics with the average value over all runs across all game parameter configurations for each model.

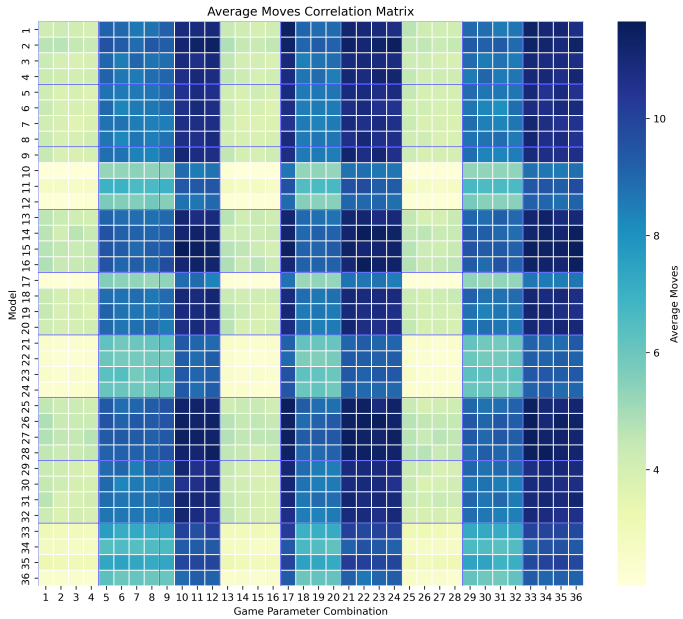

Fig. 2. Heatmap correlating models and game parameter configurations against the amount of average moves taken by an agent before termination.

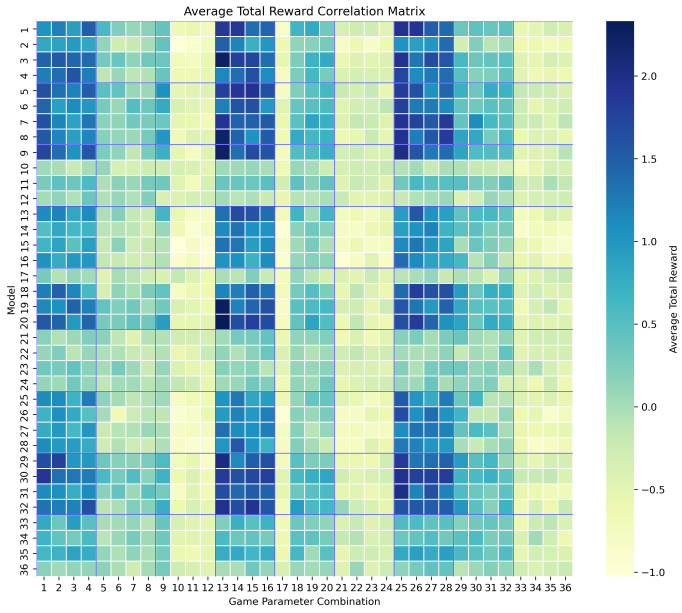

Fig. 3. Heatmap correlating models and game parameter configurations against the amount of total reward taken by an agent before termination.

the case and that they did not have a noticeable impact on the training nor the performance of models in the environments.

The existence of walls, no matter the amount of walls, did reveal an impact on the training and difficulty of the models and game environments. Model 9 and 17 highlight this as they perform vastly different compared the models with the otherwise same values. Model 9 has a map size of 19×19 and the lowest amount of max food possible on the map, but does not include the existence of walls. Both the model and game configuration was expected to perform poorly due to the map size and low amount of food when seeing the results of other models, but it was shown to yield substantially better results compared to any other environment and model with a map size of 19×19, despite there being more food on other configurations. Model 17 performed significantly worse than it's related models, but curiously the corresponding game environment itself yielded much better results for other models not trained on it compared to environments related to it. It may be that the environment configuration 17 strikes a perfect balance for the other trained models as it's map size and food amount is medium and does not have walls, thereby providing better results for them despite not being trained on it. However why the model trained on this map does not adapt well to other environments is not clear.

The bar chart in fig. 1 also suggests that the primary parameter that decides the performance of a model is the map size that it is trained on. Besides map size, there are other, less predictable parameters influencing performance. One instance is model 11 with a max wall parameter of 10 which notably out performs adjacent models 10 and 12 which respectively have wall parameters 5 and 15. This is peculiar since other models with a max wall parameter of 10 do not have the same level of superiority over their neighboring models the same way model 11 has. This inconsistency invites further investigation since the cause is not obvious.

The metrics shown in fig. 1 and along with heatmaps fig. 2, fig. 3 could be further refined by separating the winning and losing snake or by developing a metric that directly compares them. This refinement could address the issue of the losing snake negatively impacting the performance of the winning snake.

The poor performance of models trained on a map size of 19×19 and the poor adaptability of models that act in such an environment without being trained on it may be explained by the increased complexity. As space for possible actions is much greater it may have been more difficult for the agent training and any agent attempting to adapt to the environment to handle it in an adequate way. Such a problem might be solved by training the models more, but attempting to adapt models trained on lower map sizes to greater map sizes could still yield problems as it does not have the possibility of learning the increased complex scenarios.

As such it would be suitable to attempt to further this research by performing an experiment focusing on map size where the models are trained longer to attempt to verify the relationship between adaptability of agents and the different map sizes they have been trained in.

Moreover, improving the observation space by increasing or changing the amount of fields analyzed might also yield improvements in case it is limiting it's viability with a larger space. Because it may be that it's too difficult for the model to navigate and find food within the large map, so increasing the range of it's perception might yield improvements overall.

Another way to improve the performance would be to experiment with different hyperparameters. For this study the hyperparameters were not exhaustively compared, as such they might not be optimal and further refinement might yield improvements that could result in the agents trained on a more complex game setting being more successful and able to adapt to more environments.

VI. Conclusion

The study provides insights into how agents trained in differing difficult environments, that may hinder learning, impact the adaptability of agents. It can be used to support the development of more versatile agents by factoring in these results and highlighting that some parameters may greatly affect the adaptability, while others may not substantially affect the results at all.

To thoroughly explore the impact of varying game parameters on adaptability, we devised 36 different configurations for the 1v1 Snake game and trained a model on each of these environments. These models were then tested on each of the different game parameters configurations resulting in metrics that show the performance and adaptability of each.

This data was analyzed and the results gave insights into the performance and effect different game parameter contributions had on the training and adaptability of different models in different environments. From the used parameters it was found that the map size was the most significant attribute in affecting performance and the adaptability of agents. A larger map size resulted in a poorer agent and was harder for agents not trained on the environment to adapt to. The max amount of food and walls was not found to impact the training nor the adaptability of agents, but the existence of walls did negatively contribute to to the adaptability and training of agents.

The best performing agents were agents that were trained in a small map size with no walls present, however the most adaptable agents were the ones trained in a small or medium map size with walls enabled. This result is not completely unexpected, but it was thought that the higher map size would provide with a more adaptable agent. However, this may be a result of insufficient training and this could be explored in future work where the training size is increased and the effects of different map sizes is explored in greater depth. Moreover, hyperparameters were not exhaustively tuned for the experiment and as such might result in better trained agents if these were to be examined more closely.

Furthermore the metrics used for assessing the performance could be improved as they might have skewed the results negatively due to being too simple and not considering winning or losing snakes, but taking averages over both. In future work

it may be suitable to revisit this and explore different metrics for greater insight into the adaptability of agents.

REFERENCES

[1] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm,"

[2] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, and R. Tsing, "StarCraft II: A New Challenge for Reinforcement Learning," Aug. 2017. arXiv:1708.04782 [cs].

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," July 2017.

[4] J. Chung, A. Luo, X. Raffin, and S. Perry, "Battlesnake challenge: A multi-agent reinforcement learning playground with human-in-the-loop," 2020.

[5] R. Sutton and A. Barto, *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series, MIT Press, 2018.

[6] A. Finnson and V. Molno, "Deep Reinforcement Learning for Snake,"

[7] "Autonomous Agents in Snake Game via Deep Reinforcement Learning | IEEE Conference Publication | IEEE Xplore."

[8] P. Yuhang, S. Yiqi, M. Qianli, G. Bowen, D. Junyi, and T. Zijun, "Playing the Snake Game with Reinforcement Learning," *Cambridge Explorations in Arts and Sciences*, vol. 1, July 2023.

[9] 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.

[10] 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.