# Investigating the Effects of Environment Size for Co-Evolving Deep Q-Learning Agents in 'Hunters vs Runners' (Using MESA)

**Matthew A. Ford**

Department of Informatics, University of Sussex, BN1 9QH, UK

mf472@sussex.ac.uk

*Abstract* - This paper explores how the size of an environment affects the behaviours and optimisation of co-evolving agents in 'Hunters vs Runners'. Using deep Q-learning (DQL), the agents are trained to compete within a shared environment of varying sizes. This study investigates how the environment size impact the fitness and activation values of the agents to outline if they find multiple strategies, and if that is an optimal solution.

The results of this investigation suggest that, for this game, starting the agents further from their outlined goal increases performance as it allows for the system to train and traverse the fitness landscape for more time, therefore bypassing several local optima. The results offer an insight into the importance of environment design in the training of multi-agent reinforcement learning (MARL) systems.

## 1. Introduction

This paper investigates how the environment size affects the behaviour of co-evolving agents in the game 'Hunters vs Runners'.

'Hunters vs Runners' is a game where hunters attempt to stop runners from reaching the other side of the play area. In this iteration, when a runner is caught, they themselves become a hunter, this creates a dynamic shift in the game's balance.

Agent-based modelling (ABM) is common in simulating games for network studies but is more-so common in studying population dynamics *(McLane, A. et al)* and behavioural patterns. As the game of 'Hunters vs Runners' very closely resembles a predator-prey dynamic this helps to give reason to features that either agent may have.

Deep Q-learning (DQL) has had some notable successes, one similar example being deep model-free Q-learning for general Atari gameplaying *(Mnih et al.)*.

DQL is a reinforcement learning algorithm that follows an action-reward approach. Agents receive rewards, either positive or negative, based on their current state withing the environment, and their chosen action. At each iteration, a mini batch of states, actions, rewards, and next states, are sampled from the memory to train the network which then can approximate the action-value function *(Fan, J, et al.)*. This allows for some features to be emphasised over others. When rewards are well defined, this leads to the agents learning the desired behaviours.

Agent fitness is measured proportionally to their reward. Genetic algorithms help to find optimal solutions by replacing low-scoring agents with those that scored higher. This encourages the evolution of optimal strategies and adopts a natural selection *(Darwin, C)* based approach that can be observed in real-life environments where there is competition. To have an effective search there must be a fitness function and selection pressure that gives agents with a higher fitness a higher chance of being selected for reproduction *(Lozano, M)*.

As far as reinforcement-based learning is concerned, it seems that the environment is often an overlooked factor as to why agents exhibit certain behaviours (desired or not). This study hopes to give some evidence as to why this may be an effector to the overall system.

## 2. Methods

This section discusses how the experiment was designed and carried out. Any set parameters are outlined. First, the setup of the environment to contain the agents is covered, followed by the player agents and their individual network and design choices. Then, the process of fitness landscape traversal and generational replacement is discussed.

### 2.1 | Environment and Visualisation

The environment for this experiment is implemented as a class derived from from MESA's 'model' class. This allows for the scheduling and control of all agents contained within while also providing a grid to give the agents a spatial representation that is vital for interaction.

To ensure there were no biases with agent scheduling order, a random activation was used. With this, agents are chosen in a random order to decide their moves instead of being chosen in their order within the scheduler. As this is a competitive game, this is important to avoid as it would make the system unbalanced between agents and could potentially become too predictable.

The simulation environment, as depicted in *Figure 2.1,* there are five main components: hunter and runner agents, borders, obstacles, and an escape zone. The hunters and runners are dynamic agents, whereas the border, and obstacles remain static.

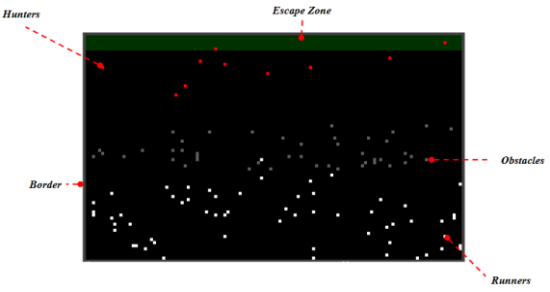


*Figure 2.1: A capture of the game environment, labelled with its key features.*

To establish the initial conditions, the hunters, runners and obstacles are randomly positioned within the top, centre, and bottom 20% of the environment's height, respectively. The choice to use these randomised spawn points helped reduce the chance of overfitting the system, ensuring the hunters and runners do not learn fixed paths from a predetermined starting point and that obstacle arrangements do not become learnt by the system, therefore preventing the system from replicating the same sequence of actions over multiple games.

As the parameter sweep increases the width and height, the initial setup consists of 5 hunters, 30 runners, and 20 obstacles. To prevent having excessive sparsity in the larger environments, these quantities are proportionally scaled with the increase in width while remaining a fixed ratio of 1:6:5.

Each game continues until either there are no more active runners, or the step count exceeds 10,000. This threshold allows enough time for the networks to explore, learn, and optimise their decision-making processes. Each match consists of 11 games, with a population regeneration every 4.

### 2.2 | Player(s)

Each player operates with their own, independent neural network to determine their next actions. The structure of these networks is consistent across both teams, with three hidden layers between the input and output. To enhance the training stability, layer normalisation is first applied between the layers. Leaky ReLU activation functions are used to prevent neurons becoming inactive at zero-values, and dropout is applied at a rate of 15% to all but the final hidden layer to reduce the chance of overfitting.

*Figures 2.2*, and *2.3* show the network diagrams for the hunters and runners. The only distinction between these networks is the increased output size for the runners which is due to them having more possible actions

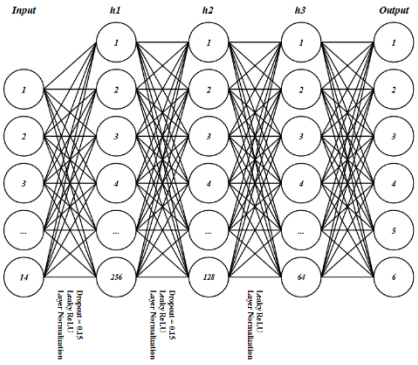


*Figure 2.2: A visual representation of the hunter neural network*

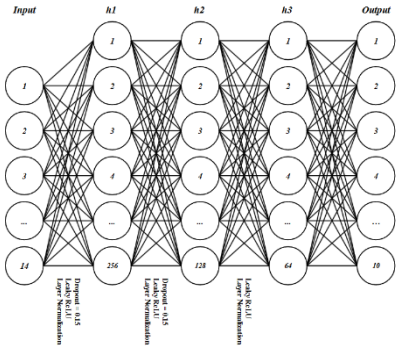


*Figure 2.3: A visual representation of the runner neural network*

The reason for runners having more actions than hunters is due to what is seen in natural systems. It is more common for prey to have movement advantages over predators, something which has been incorporated in this simulation.

As illustrated in *Figure 2.4* the runners have the ability to move diagonally, making their movement on the grid more efficient which is useful for evading hunters. To further constrain the hunter movement, it was decided that it will only be possible for them to change direction every three steps. This limitation prevents the hunter from making turns that were too sharp, ensuring each move was more deliberate and strategic.

| | | |
|---|---|---|
| *R* | *H & R* | *R* |
| *H & R* | ***Agent*** | *H & R* |
| *R* | *H & R* | *R* |

*Figure 2.4: Possible movement directions of hunter (H) and runner (R) agents at any given time.*

To limit the 'sight' distance of the players, a decision was made to limit their vision to a neighbourhood size of 10. This is again a Moore neighbourhood structure and therefore forms square, spanning 10 grid spaces from all sides of the agent.

For training, each agent network receives a constructed state representation. As depicted in *Figure 2.5*, states consist of 14 numerical values. The decision to include only the relative position of the nearest enemy agent was an intentional design choice to restrict the data entering through the state and preventing unnecessary complexity. By limiting this information, agents are encouraged to develop direct strategic approaches, making them more so react to imminent threats or opportunities.

| *Initial Team* | *Current Team* | *Relative Position* | *Current Position* | *Closest Hunter Position* | *Reward* |
|---|---|---|---|---|---|
| *0* | *0* | *[1, 0]* | *[26, 40]* | *[5, 3]* | |
| ***Nearby Hunter Count*** | ***Can Catch Runner*** | ***Steps Survived*** | ***Has Escaped*** | ***Last Move*** | *35* |
| *2* | *0* | *24* | *0* | *6* | |

*Figure 2.5: An example of a state representation memorised and processed by the network to train and make action predictions more accurate.*

These states are stored in a memory backlog which is used in training the neural network to extract features / trends with the states and their given reward.

### 2.2.1 | Hunters

The rewards for the hunters focus on moving towards and successfully capturing runners. *Figure 2.6* presents the reward table for the outlined actions.

| ***Situation*** | ***Reward*** |
|---|---|
| *Stay Idle* | *-3 * consecutive idle steps* |
| *New Position Visited* | *+3 if True, else -2* |
| *Against Border* | *-50* |
| *Move Closer to Runner* | *+5* |
| *Keep Distance from Runner* | *- 2* |
| *Move Away from Runner* | *-10* |
| *Catch Runner* | *+1000* |
| *Was a Runner* | *+ve reward *= ¼* |

*Figure 2.6: The reward table for agents on the hunter team.*

The structure of these rewards incentivises hunters to maintain movement, avoid borders, and pursue and successfully catch runners.

As the agent must choose to catch the agent through a network output, they will slowly learn what is the viable circumstance to catch a runner. As the runners can be caught and converted to a hunter, to prevent exploitation, any positive reward given to these agents are quartered. This is to discourage runners from deliberately seeking capture to increase their reward as a hunter.

An element of the state representation seen in Figure 2.5 is the "can catch runner" input. To prevent newly converted hunters from immediately catching their neighbours, there is a 20-step cooldown to give time for runners to move.

In addition to their existing constraints, the hunters also are restricted from switching direction immediately. Instead, they can only change their trajectory every 3 steps. This makes it so that a hunters movements need to be more strategic to intercept runners. Once again, this mimics what would be seen naturally with runners (prey) being more agile than hunters (predators).

### 2.2.2 | Runners

The main goal of the runners is to avoid the hunters and make it to the other side of the play area. *Figure 2.7* shows the reward table used for this.

| *Situation* | *Reward* |
|---|---|
| *Stay Idle* | *-3 * consecutive idle steps* |
| *New Position Visited* | *+3 if True, else -2* |
| *Against Border* | *-50* |
| *Escaped* | *+10,000* |
| *Enemy Near + Grouped* | *- number of teammates nearby* |
| *Move Closer to Hunter* | *-2* |
| *Keep Distance from Hunter* | *-1* |
| *Move Further from Hunter* | *+5* |
| *Move Towards Goal* | *+1* |
| *Caught by Hunter* | *-1500* |

*Figure 2.7: The reward table for agents on the runner team.*

As seen by comparing the two reward tables, the entries for staying idle, visiting new positions, and staying away from borders are the same. This is because that specific behaviour is required to stop agents from gathering in corners or against borders which was very commonly seen in the early stages.

The reward for escaping is much larger than any of the other rewards as this is the main goal of the runners, and once achieved, the runner cannot earn any more points for the game. Therefore, this acts as compensation.

To discourage groupings of runners when there is an enemy within sight distance, a negative reward is added when an enemy can be seen, and a teammate is directly next to the agent.

When a runner is caught by a hunter, it also adds another substantial negative reward. This is to highly discourage this behaviour, and through the networks ability to decipher patterns, this should allow the runners to gradually learn to evade hunters. This therefore will lead to an increase in escapees.

## 2.3 | Genetic Algorithms and Agent Metrics

To define a fitness measure for the performance of agents, a continuous reward was retained across multiple games. As the system is based on rewards, this meant the fitness was proportional to the reward.

$$\boldsymbol{fitness \propto reward}$$

To traverse the fitness landscape and identify optimal activation values and network weights, a simulated annealing-inspired approach was employed. Here, exploration rate ($\varepsilon$) was equivalent to the 'temperature'. The decay of $\varepsilon$ over time is controlled by a decay rate ($\lambda$) where:

$$\boldsymbol{\varepsilon_{t+1} = \varepsilon_t * \lambda}$$

As seen in *Figure 2.8*, for each game, the exploration started at 1 meaning that the moves were decided completely randomly. Then this was decreased every step to help the system converge on an optimal solution.

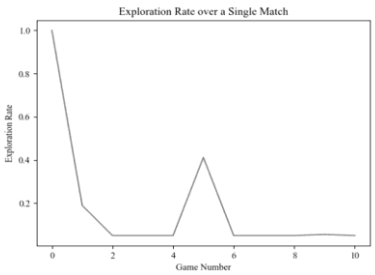


*Figure 2.8: The decrease of the exploration rate over the course of a match.*

To help prevent the training converging on local optima, the exploration rate increases by 10% every time it is detected that the average reward is decreasing. This allows for the searching algorithm to make 'jumps' and hopefully explore new moves from its current position that increase the reward again. This can be seen in the 5th game.

Other hyperparameters that were also controlled included learning rate and discount factor. Learning rate was again decayed over time using the same decay rate as the exploration. This helped reduce the step sizes when the network set new q-values and ultimately allowed for a faster convergence.

Discount factor was used to prevent cyclic behaviour where agents find repeated sequences of movements to achieve a high reward without following the desired behaviour. This meant that the reward given by actions was decreased by a small constant over time.

For population management, a simple truncation selection function was used, tracking the top three preforming agents. The lowest performing 50% of agents were replaced using a hybrid selection approach. As illustrated in *Figure 2.9*, 10% were replaced with a completely random selection from the top three agents of that type, and the remaining 40% were replaced with a random choice of all agents in the top 50% with the selection probabilities weighted by reward. This gave a 'survival of the fittest' dynamic whilst preserving population diversity and avoiding convergence on an early local optimum.

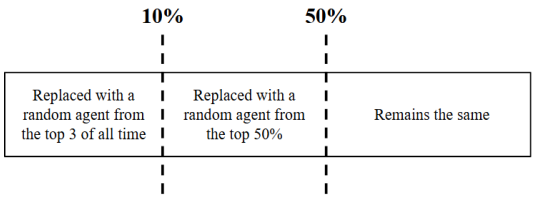


*Figure 2.9: A representation of the truncation selection process of replacing the population of agents.*

## 3. Results

This section discusses the results produced from the parameter sweep. First identifying the highest and lowest performing values followed by an exploration into the agent 's fitness and activation values for each.

*Figure 3.1* shows the heatmap of the percentage of escaped runners for each value of the parameter sweep. Here there is a clear gradient showing that a larger height was more influential to the performance than the width.

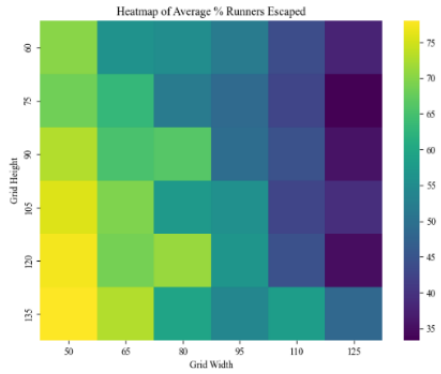


*Figure 3.1: A heatmap showing the average percentage of hunters escaped for each match run during the parameter sweep.*

This figure shows that the highest performance is at:

$$w = 50 \; and \; h = 135$$

And the lowest performance is at:

$$w = 125 \; and \; h = 75$$

### 3.1 | Highest Performance Results

As seen in *Figure 3.2*, the runner's average consistently stays at ~0. This is much higher than the top performing hunters. It must be noted that on these graphs, the default value for there being no current top hunter / runner is 0. Therefore, all the fitness graphs start from here. As the top agent averages are so far apart, this

means that the runners are completing their goals more often than hunters.

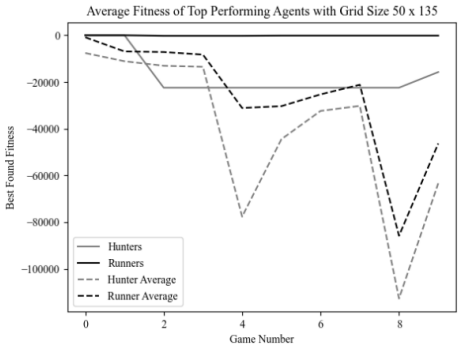


*Figure 3.2: A graph showing how the average and best finesses of the hunter and runner agents change each game for the best performing match.*

*Figure 3.3* shows the average activation values of each agent's network layers over the course of a match. As seen, for both the hunters and the runners, the "*X1*" and "*X2*" layers both have a lot of noise but are gradually growing upwards.

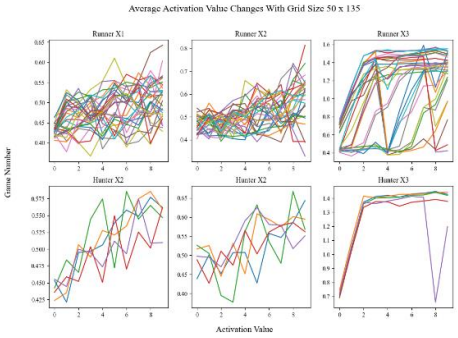


*Figure 3.3: Plotting of the average activation values on all three layers as the match progresses for the highest performing match.*

When it comes to the "*X3*" layer, it shows that most of the agents fall for a higher activation value. This shows that the agents are acting more aggressively with the feature in this layer. As there aren't more than one obvious cluster, it can be said that in this instance, all the agents learnt to adopt the same strategy.

As the gradient also converges, this also proves that the model was starting to stabilise and therefore shows that overfitting was avoided.

## 3.2 | Lowest Performance Results

*Figure 3.3* shows the average fitness for the lowest performing match.

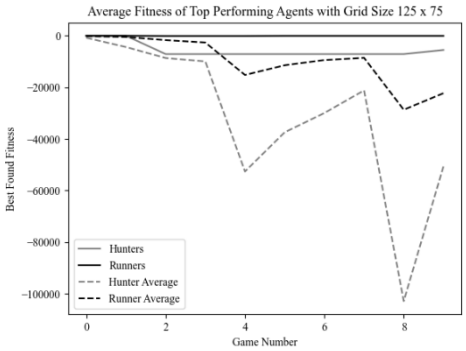


*Figure 3.4: A graph showing how the average and best finesses of the hunter and runner agents change each game for the worst performing match.*

Compared to the best performing match, the top agent's averages are much closer together. This would have therefore caused the hunters to be more effective in catching the runners and as a result, the match had a lower escape percentage. In terms of the overall averages for each team, they don't deviate too far from that shown in *Figure 3.2*. This therefore shows that the training process per match is very similar every time, just different local optima are found.

Looking into the activation values shown in *Figure 3.5,* the first two layers' activation values have a gradual increase again. The more noticeable difference is in the "X3" layer values. The hunters again converge on a higher activation value, therefore becoming more aggressive on that feature, but the runners have two main clusters. It seems that they are split between having a high and low activation value.

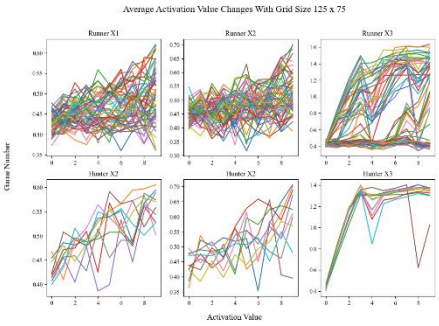


*Figure 3.5: Plotting of the average activation values on all three layers as the match progresses for the lowest performing match.*

Due to the goal being very near, this split could have come from the agents reaching the goal too quickly and therefore did not have the time to explore fitness landscape more thoroughly. But for the hunters, this allowed them to efficiently reach agents nearby and complete their goal, making it a much more optimal scenario for them.

### 3.4 | Further Analysis of Results

Given the data provided from this experiment, and upon visualising the results; to have a higher escape rate, it was found to be more optimal for runners to start further away from the goal and therefore more important to have a larger environment height.

The reasoning behind this is that the agents have more time to learn and converge on similar optimal solutions. This is seen in the activation values where the agent's averages were all the same range. Having games end too quickly, possibly due to random moves, has seemed to change the population strategies and made some agents converge on a less-fit local optima which is undesirable.

## 4. Discussion

In conclusion, this experiment has shown that for similar games to 'Hunters vs Runners', it is very important for the environment to allow for space between agents and their goals. By limiting the environment size, it allows for agents to get stuck on local optima which can be avoided given exploration rate changes and time to process and learn from the new actions taken.

Regarding the hypothesis, this has proven it to be incorrect as it appears that having a goal be discovered too early affects the development of these types of agents.

The system itself does prove to be adaptive as co-evolution forces both the runners and hunters to react to each other's strategies to maximise their reward.

The next step of this report would be to explore the network sizes and see how that may affect the overall performance by allowing agents to define more features and have a much more granular analysis of the current state.

This report hopes to give some contribution to the field of agent-based modelling in respect to the design of the environment dependent on the overall goals.

## 4. References


**Darwin, C (1859)** On the Origin of Species.

*https://www.taylorfrancis.com/chapters/edit/10.4324/9781003194651-10/origin-species-charles-darwin*

**Fan, J., Wang, Z., Xie, Y., Yang, Z.** (2020) A Theoretical Analysis of Deep Q-Learning.

https://proceedings.mlr.press/v120/yang20a

**Lozano, M. Herrera, F. Cano, J. R.** (2008). Replacement Strategies to Preserve Useful Diversity in Steady-State Genetic Algorithms

https://doi.org/10.1016/j.ins.2008.07.031

***McLane, A, Semeniuk, C, McDermid, G, Marceau, D.*** *(2011). The role of agent-based models in wildlife ecology and management.*

*https://doi.org/10.1016/j.ecolmodel.2011.01.02*

***Mnih, V., Kavukcuoglu, K., Silver, D. et al****. (2015). Human-level control through deep reinforcement learning.*

*https://doi.org/10.1038/nature14236*