# Battlesnake Challenge: A Multi-agent Reinforcement Learning Playground with Human-in-the-loop

**Jonathan Chung**[*1] **Anna Luo**[*1] **Xavier Raffin**[1] **Scott Perry**[1]

## Abstract

We present the Battlesnake Challenge, a framework for multi-agent reinforcement learning with Human-In-the-Loop Learning (HILL). It is developed upon *Battlesnake*, a multiplayer extension of the traditional *Snake* game in which 2 or more snakes compete for the final survival. The Battlesnake Challenge consists of an offline module for model training and an online module for live competitions. We develop a simulated game environment for the offline multi-agent model training and identify a set of baseline heuristics that can be instilled to improve learning. Our framework is *agent-agnostic* and *heuristics-agnostic* such that researchers can design their own algorithms, train their models, and demonstrate in the online Battlesnake competition. We validate the framework and baseline heuristics with our preliminary experiments. Our results show that agents with the proposed HILL methods consistently outperform agents without HILL. Besides, heuristics of reward manipulation had the best performance in the online competition. We open source our framework at https://github.com/awslabs/sagemaker-battlesnake-ai.

## 1. Introduction

Battlesnake is an extension of the traditional *Snake* arcade game where multiple snakes compete against one another for food and survival. The last surviving snake is the winner of the game. Competitors traditionally develop heuristics such as using the A* search algorithm (Russell & Norvig, 2002) and the tree search (Schier & Wüstenbecker, 2019) to seek food, enemy heads, and its tail. Meanwhile, Reinforcement Learning (RL), which learns a policy by interacting with an environment through trial-and-error, has been naturally adopted to tackle such sequential problems. Recent advances in deep RL further allows modelling such decision making problems with high-dimensional visual perceptual inputs made up of thousands of pixels (Mnih et al., 2015). Battlesnake focuses on a particular branch of RL where multiple agents learn to interact within the same environment. It is denoted as multi-agent RL problems (Littman, 1994; Bu et al., 2008; Buşoniu et al., 2010). The game fits the fully competitive setting (Silver et al., 2017) in which each agent is tasked to maximise its own reward while minimising their opponent's rewards.

Developers with superior domain knowledge build snakes with unique strategies and heuristics. Having Human-In-the-Loop Learning (HILL) aids the policy training and prevents catastrophic actions (Christiano et al., 2017; Abel et al., 2017). Human intuition could be provided as feedback (Arakawa et al., 2018; Xiao et al., 2020), teachers (Abel et al., 2017; Zhang et al., 2019b), and overseers (Saunders et al., 2018). It can also steer agents to have more optimal learning by identifying important and omitting misleading features, subsequently reducing the dimensionality of the state space (Abel et al., 2017). However, these methods incorporate human guidance in a single agent setting. To the best of our knowledge, there exists no playground designated to evaluate multi-agent RL algorithms with HILL.

To fill in this gap, we introduce the Battlesnake Challenge, an accessible and standardised framework that allows researchers to effectively train and evaluate their multi-agent RL algorithms with various HILL methods. We choose to use Battlesnake as the underlying game engine because it can be seamlessly integrated to the multi-agent RL research direction, while remaining intuitive and lightweight. Besides, the rules governing the game are relatively simple, but the resulting strategies can still be complex. Such setting facilitates the use of human knowledge to provide policy training guidance. In specific, we offer a simulated Battlesnake environment for offline training, after which the snakes can be deployed to the cloud to compete with other snakes in the Battlesnake global arena[1]. To accommo-

[*]Equal contribution [1]Amazon Web Services. Correspondence to: jonchung, annaluo <@amazon.com>.

*2nd ICML 2020 Workshop on Human in the Loop Learning*, Vienna, Austria, PMLR 108, 2020. Copyright 2020 by the author(s).

[1]https://play.battlesnake.com/arena/global/. The arena hosts all types of AI bots, being RL or non-RL, to compete against each other.

date human knowledge, we identify a number of standard heuristics and further demonstrate the baseline techniques to incorporate them into the agent. Our proposed framework is *agent-agnostic* and *heuristics-agnostic* such that researchers can design their own algorithms, train their models, and demonstrate in the real Battlesnake competition. We hope that the Battlesnake Challenge will serve as a testbed to encourage research into multi-agent RL and HILL. Our contributions are as follows:

- We propose an end-to-end training-deployment-testing framework that consists of an offline module for multi-agent RL training with HILL, and an online module for performance evaluation against other public agents.
- On the multi-agent RL aspect, we develop a simulator for the Battlesnake problem through a proper design of state, action and reward. On HILL aspect, we identify a set of baseline heuristics that can be instilled to improve the agents during training or after deployment.
- We validate the proposed framework and baseline heuristics with our preliminary experiments. We investigate how different incorporation methods affect task performance both offline and online. Our results show that agents with HILL outperform agents without HILL and careful reward manipulation performs the best among our proposed heuristics in the online Battlesnake arena.
- We open-source the Battlesnake Challenge framework at `http://github.com/awslabs/sagemaker-battlesnake-ai` to encourage broader research directions.

## 2. Related works

**Multi-agent RL Testbed:** There is a growing number of studies focusing on designing environments to evaluate the agents' performance with the advancements in the multi-agent RL regime (Zhang et al., 2019a; Nguyen et al., 2020). For example, Keepaway soccer (Stone et al., 2005) and its extension (Kalyanakrishnan et al., 2006; Hausknecht et al., 2016) provide a simulated football environment. Meanwhile, a set of gridwold-like environments has been developed to encompass various multi-agent tasks, covering both continuous (Lowe et al., 2017) and discrete (Yang et al., 2018; Zheng et al., 2018) control problems. Resnick et al. (2018) proposed *Pommerman*, a game stylistically similar to the Nintendo game Bomberman, as a playground for benchmarking different agents. The system uses low dimensional symbolic state interpretations input, and the authors built an online leader board where researchers could submit their agents and compete against one another. It, however, allows only up to four agents and does not include the mechanism that adds human intuition to the agents.

More recent work considers real-time strategy games that require complex environments and controls. The StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019; Vinyals et al., 2019), developed based on *StarCraft II*, focuses on handling partially observability and high-dimensional inputs. It aims to serve as a benchmark for *cooperative* multi-agent RL, rather than the *competitive* setting as in Battlesnake. In contrast, Ye et al. (2019) presented a 1v1 game mode using multi-agent RL in a competitive setting through the game *Honor of Kings*. They developed an off-policy RL system architecture for scalable training and eliminated invalid actions to improve the training efficiency. Nonetheless, the complexities inherent in both of these games require specific game knowledge, making it difficult for general researchers to develop and evaluate their multi-agent RL algorithms.

**Human-in-the-loop Reinforcement Learning:** The data hungry nature of RL has prompted researchers to develop techniques to leverage human knowledge for RL tasks (Zhang et al., 2019b). Often, human information is passed along in the form of human intervention (Saunders et al., 2018), reward shaping (Knox & Stone, 2008; 2009; 2012; Warnell et al., 2018; Arakawa et al., 2018; Xiao et al., 2020) and policy evaluation (Griffith et al., 2013; MacGlashan et al., 2017; Arumugam et al., 2019). In principal, human intuition could be injected in two methods, 1) by evaluating the actions during training through real-time feedback or intervention; and 2) by defining handcrafted rules prior to training to alter the agent's behaviour based on human intuition. TAMER+RL (Knox & Stone, 2010) is an example of the first method in which human provide a reward given a state action pair. An example of the second method is the agent-agnostic framework proposed by Abel et al. (2017). The framework contains a protocol program that controls the human interaction with a single agent. Human can perform learning interventions with handcrafted rules that alter the transition dynamics given the current state and action, with methods including action masking, reward manipulation, etc. Action masking is a technique to bar possible actions from the policy based on engineered rules. Reward manipulation is the act of specifically designing a reward function. In particular, the authors found that action masking simplifies the RL problem and was effective to prevent catastrophic actions (Saunders et al., 2018). Considering that Battlesnakes are traditionally developed with handcrafted rules, we extend the framework into a multi-agent RL setting.

## 3. Description of the Battlesnake Challenge

The architecture design of the Battlesnake Challenge is presented in Figure 1. Our framework includes an offline RL training module equipped with a simulated Battlesnake environment, as will be described in Section 3.1. The framework is designed to work with handcrafted rules (Abel et al., 2017) as well as more complex heuristics (Russell & Norvig, 2002; Schier & Wüstenbecker, 2019) that are programmed in advance. The preprogrammed heuristics are independent of the RL algorithm and could be incorporated into the RL training module to assist the training procedures.

Once the trained agent is deployed online, it can make inferences and interface with the Battlesnake engine to play in the Battlesnake arena. We integrate the inference with optional ad-hoc heuristics to enable action overwriting, through which human experts can perform the safety checks. We use the Battlesnake arena to evaluate different combinations of in-training and ad-hoc human guidance by allowing the agents to compete against each other.

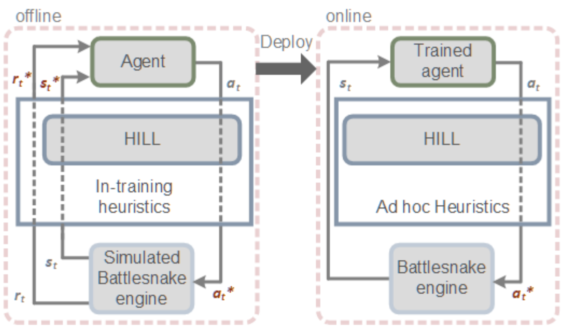

*Figure 1.* Overview of the Battlesnake Challenge with human-in-the-loop multi-agent reinforcement learning. Human knowledge injected in the offline training phase can possibly affect state ($s_t$), action ($a_t$) and reward ($r_t$) at timestep $t$. Once the agents is deployed, Ad-hoc heuristics can overwrite the action ($a_t$) at each inference timestep $t$.

## 3.1. Battlesnake description

We first provide a detailed description of Battlesnake game logic. A typical game in Battlesnake consists of three to five snakes on a board ranging from $7 \times 7$ (small), $11 \times 11$ (medium) to $19 \times 19$ (large). At the start of the game, $N$ snakes are randomly distributed along the boundaries of the board, each with health of 100. There is one piece of food randomly distributed at the same time. At each turn, the health of every snake decreases by one and each snake reacts to the environment indicating whether it will move up, down, left or right; food are then randomly spawned. Unlike the traditional snakes game, if a snake is facing up and the next action is to move down, the game considers the snake hitting its own body and it will be eliminated from the game. This is known as a *forbidden move*. If a snake eats a food, its health will be returned to 100 and its length will grow by one. If a snake hits another snake's head, the shorter of the two snakes is eliminated from the game (if the sizes of the two snakes are the same, both the snakes are eliminated). This is referred to as *eating another snake*. In addition, a snake is eliminated from the game if it: 1) goes out of the boundaries of the map, 2) hits another snake's body, 3) hits its own body, or 4) has a health of 0. The last surviving snake becomes the winner.

## 3.2. Battlesnake as a Reinforcement Learning Environment

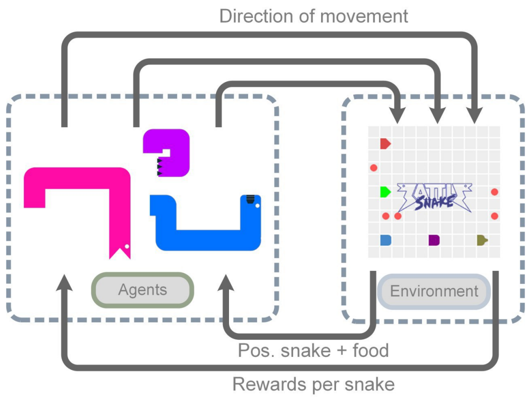

*Figure 2.* Modelling Battlesnake with reinforcement learning.

We consider a standard Markov game (Littman, 1994) to model the interaction between multiple Battlesnake agents with the environment. Each agent represents one snake in the game. The Markov game is specified by a tuple $M = (\mathcal{N}, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\mathcal{N}}, \mathcal{T}, \{R^i\}_{i\in\mathcal{N}}, \gamma)$, where $\mathcal{N} = \{1, \dots, N\}, N > 1$ denotes the set of agents, $\mathcal{S}$ is the state space observed by all agents and $\mathcal{A}^i$ is the action space of agent $i$. $\mathcal{T} : \mathcal{S} \times \mathcal{A}^1 \times \cdots \mathcal{A}^N \times \mathcal{S} \to [0, 1]$ denotes the transition function that maps a state $s_t \in \mathcal{S}$ and action $a^i_t \in \mathcal{A}^i$ pair for each agent $i$ to a probability distribution over the next state $s_{t+1} \in \mathcal{S}$. The environment emits a reward $R^i : \mathcal{S} \times \mathcal{A}^1 \times \cdots \mathcal{A}^N \to \mathbb{R}$ on each transition for each agent $i$; $\gamma$ denotes the discount factor. Figure 2 illustrates setup in which the agent interacts with the simulator over the OpenAI Gym interface (Brockman et al., 2016). Components in the Markov game are given as follows:

**State:** We provide the Battlesnake simulator a gridworld based state space $s_t$ at time $t$ to represent the spatial distribution of all the snakes and food. Agent $i$ is represented by a list of coordinates, $\mathbf{x}^i = [x^i_1, x^i_2, \dots, x^i_{L_i}]$ where $x^i_j \in \mathbb{R}^2\ \forall j \in 1, \dots, L_i$ and $L_i$ is the length of snake $i$. The $N$ snakes are collectively referred to with $\mathbf{X}$ such that $\mathbf{X} = [\mathbf{x}^1 ... \mathbf{x}^i ... \mathbf{x}^N]$. Food $\mathbf{F}$ is represented as $[y^1, y^2 \dots y^{M^t}]$ where $y^j \in \mathbb{R}^2$ is the coordinate corresponding to the location of the food $j$ and $M^t$ represents the number of food at time $t$. To ingest the information of other agents, we organise the state for agent $i$, $s^i_t$ as a grid with 3 channels where $s^i_t \in \mathbb{R}^{w \times h \times 3}$, and $w$ and $h$ are the width and height of the map. Specifically, assume $s^i_t[j, k, c]$ describes the state value at coordinate $(j, k)$ on the map of channel $c$ for agent $i$ at time $t$. Channels $c \in \{0, 1, 2\}$ represents position of food, agent $i$, and other agents, respectively. For $c = 0$, $s^i_t[j, k, 0] = 1$ if $(j, k) \in \mathbf{F}$ and 0 otherwise. The state for $c = 0$ should be identical for all agents. Similarly, $c = 1$ provides the position of agent $i$ where $s^i_t[j, k, 1] = 1, \forall (j, k) \in \mathbf{x}^i$ and 0 otherwise. Also, we set $s^i_t[j_h, k_h, 1] = 5$ where $(j_h, k_h)$ denotes the head of agent $i$. We choose 5 experimentally to provide a larger differentiation between the body and the head of the snake. Finally, $c = 2$ is defined as $s^i_t[j, k, 2] = 1, \forall (j, k) \in \mathbf{x}^{i'}$ where $\mathbf{x}^{i'}$ are all other agents in $\mathbf{X}$ where $i' \neq i$, and the heads of the snakes in $\mathbf{x}^{i'}$ are set to 5 as well. It is worth mentioning that we deliberately choose to formulate the state space in a gridworld fashion rather than using image pixels or complex embedded features. We believe this will make the RL training easier and encourage research focus on developing the heuristics with HILL.

**Action:** The action space $\mathcal{A}^i$ is identical for each agent $i$, representing the direction the agent moves towards in the next turn. Namely, $a^i_t = [0, 1, 2, 3]$ corresponds to up, down, left, and right at time $t$ for agent $i$. Thus the joint action space is defined as $\mathbf{a}_t = [a^1_t, a^2_t ... a^N_t]$.

**Reward:** The overall goal is to become the last surviving snake. We by default apply the same reward formulation for all the agents. Specifically, a negative reward $-1$ is imposed if a snake dies, and the last snake alive is assigned with a reward 1. We grant a small reward $\epsilon = 0.002$ for each snake when it survives another turn, with an intuition to promote the snakes to move and grow.

## 3.3. Training Algorithm

We train each snake's policy independently using the Proximal Policy Optimisation (PPO) algorithm (Schulman et al., 2017). It is a widely used, modern on-policy actor-critic algorithm that has presented stable performances in many of the recent successes in deep RL. The algorithm employs two neural networks during training – a policy network and a value network. The policy network interacts with the Battlesnake environment and generates Gaussian-distributed actions given the state. The value network estimates the expected cumulative discounted reward using the generalised advantage algorithm (Schulman et al., 2015). Note that while we use PPO to conduct experiments, the proposed framework is *agent-agnostic* and can fit various discrete action-based state-of-the-art algorithms such as QMIX (Rashid et al., 2018) and SAC (Haarnoja et al., 2018).

## 3.4. Heuristics with human-in-the-loop learning

In principal, our proposed framework is *heuristic-agnostic* such that researchers can tackle various heuristic development and bring them into the agent training process. For the purpose of illustration, in this work we identify several human engineered heuristics that we used as part of the challenge. The philosophy is to provide agents information regarding what actions to *avoid* and to guide agents towards superficial skills such as heading to food when starving. Specifically, we provide the following heuristics:

1. Avoid hitting the walls.
2. Avoiding moving in the opposite direction as the snake is facing (*forbidden moves*).
3. Moving towards and eating food when the snake health is low to prevent *starving*.
4. Killing another snake (e.g., eating another snake or trapping another snake).

| Rule | Prevention/ Promotion | Interaction | Training phase |
|---|---|---|---|
| 1 | Prev. | Env. | Early |
| 2 | Prev. | Env. | Early |
| 3 | Promo. | Env. | Middle |
| 4 | Promo. | Agents | Late |

*Table 1.* Properties of the heuristics. Prev. refers to action prevention; Promo. refers to action promotion. Interaction describes whether the rules are interacting with the environment or other agents. Training phase indicates when the rules become more important.

Table 1 provides an overview of the properties of each rule. Prevention/promotion refers to *action prevention* or *action promotion*. Action prevention rules help eliminate unreasonable actions, similar to the use of catastrophic actions prevention (Abel et al., 2017; Saunders et al., 2018) and action filters (Gao et al., 2019; Meisheri et al., 2019). In most cases, action prevention rules could be resolved with a single conditional statement. Meanwhile, the described action promotion rules are more complex as they require multiple steps to achieve. Interaction describes whether the rules are interacting with the environment or other agents. For example, to manage rule 4, an agent would have to anticipate the movements of other agents in order to kill them. Finally, training phase indicates when the rules become more important. Rules 1 and 2 are necessary for basic navigation and movement; they prevent the agents from committing "suicide" in the early phases of training and are essential throughout the duration of training. Rules 3 is necessary for survival after the early phases of training when basic navigation is learnt. Rule 4 requires high level strategies once the snakes have no issues with surviving.

While human rules are instilled with a goal to accelerate the learning procedure, they can also be biased and limiting the snakes' performance. For instance, rule 3 could lead to snakes focusing too much on food, whereas rule 4 could result in over-aggressive snakes. To this end, we design our platform such that the impact of the heuristics can be controlled and even removed once an agent acquires some basic skills. Such heuristic impact break-down can also be phrased as a curriculum learning method where a logical ordering or hierarchy of simple skills is learnt during training (Matiisen et al., 2019; Portelas et al., 2019). We now describe three methods to include the heuristics rules into the RL learning.

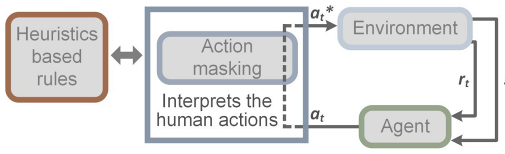

*Figure 3.* In-training action masking.

**In-training action masking:** We first consider the case where the human heuristics are injected during training and the agent adjusts its policy accordingly (Figure 3). In particular, we incorporate the feedback at the final output layers of the policy, where invalid actions are masked out of the softmax by scaling the probability to zero. The actual action $a_t^*$ taken after the masking is then passed into the simulated environment to generate the new states. Action masking applies to catastrophic action prevention and single step heuristics. As such we apply it for rules 1 and 2.

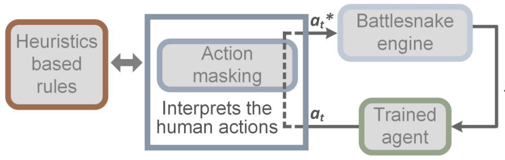

*Figure 4.* Ad-hoc action overwriting

**Ad-hoc action overwriting:** A trained agent can be deployed to interact with the Battlesnake Engine and compete with other snakes. Ad-hoc action overwriting is often applied in this scenario to enhance robustness and guarantee performance. As shown in Figure 4, it is different from in-training action masking in the sense that the heuristics are only applied during inference. In our proposed Battlesnake Challenge framework, the actual actions taken and corresponding next states are not used to update the policy in real-time as the agent is already deployed. However, the experiences can be stored for the purpose of policy evaluation.

**Reward manipulation:** Reward manipulation includes human intuition by specifically designing a reward function to encourage events corresponding to the heuristics. During training, the reward function is defined as $\hat{R}(s_t^i, a_t^i) = \hat{r}_t^i$ where $\hat{r}_t^i$ denotes the heuristics based reward. $\hat{r}_t^i$ is then fed into the learning process with the final reward function defined as $r_t^{*i} = r_t^i + \hat{r}_t^i$. For instance, we add a penalty term ($\hat{r}_t^i = -0.4$) in our experiments whenever a snake hits a wall to account for rule 1. Please note that the sign of $\hat{r}_t^i$ changes if the heuristics is action promoting.

## 4. Experiments

### 4.1. Implementation details

We open-source our proposed framework and provide implementation for each component presented in Figure 1. Specifically, our package is featured with a simulated gym environment, a heuristics developer, and the orchestration code to deploy trained agents to the Battlesnake arena. In addition, we provide a suite of training examples using the RL package RLlib (Liang et al., 2017) within the Amazon SageMaker RL package.

### 4.2. Evaluation

There are two main avenues to evaluate the RL agent with HILL. The first avenue evaluates agent performance during training. The second avenue is to use the leader board in the Battlesnake arena, in which a deployed agent competes against other snakes with black-box mechanisms.

For the first avenue, we collect the maximum *episode length* of all agents as a metric for evaluation. Episode length provides an indication of how long the agents survived for, and it's consistent across different reward manipulation schemes. We first investigate the baseline performance with map sizes of $7 \times 7$, $11 \times 11$, and $19 \times 19$ comparing the performances of training with 4, 5, and 6 agents without human intuition. We aim to verify that the our simulator is properly formulated such that the snake's performance improves over training. We then compare the performances of heuristic incorporation methods described in Section 3.4. In addition to the episode length, we also record the frequency of events that each heuristics is designed to prevent or encourage to evaluate how well the heuristics are incorporated. Here and on-wards, we fix the map size and number of agents at $11 \times 11$ and 5 respectively.

For the second avenue, we evaluate the performances of the described baseline heuristics in the arena. We wish to call out that we do not compare with the existing mature snakes. The focus of this study is to showcase how each component in our proposed framework can be modified with a minimum amount of efforts, rather than to develop a sophisticated fine-tuned agent that beats the best performing snakes. Particularly, we use rule 2 as an example to compare the performance of the trained agents with in-training action masking, ad-hoc action overwriting, reward manipulation, and vanilla training with no HILL. For each baseline heuristics, we randomly select one policy for deployment. Note that the ad-hoc action overwriting agent uses the vanilla trained agent as the base agent. For the in-training action masking agent, we apply the same masking logic during inference to ensure consistency in the state transition dynamics. We conduct two experiments for the arena testing. The first experiment consists of 30 games with the four snakes in the arena. The last surviving snake is given four points, the second last surviving snake is given three points, and so on. We also record the frequency of forbidden moves. In the second experiment, the four snakes compete in a 1 vs. 1 format. Each pair of snakes plays 10 games, leading to a total of 60 games. The winning snake gets 1 point and the losing snake gets 0 point.

## 5. Results

In this section, we present experimental results for components included in the Battlesnke Challenge. We first demonstrate the performance of the multi-agent RL agents trained in our simulated environment, and then move to performance evaluation with HILL both offline during training and online in the arena.

**Multi-agent reinforcement learning:** The results of varying the number of agents and the map size are presented in Figure 5. We train three different instances of each case with different random seeds. The solid curves correspond to the mean and the shaded region to the minimum and maximum values over the trials. We observe that after 1M steps of training, the episode length increases from 0 to an average of 175. After training for 2M steps, the growth increases steadily. We can observe that the maximum episode length for 4 agents is distinctively better than 5 and 6 agents. This is not surprising as games with 4 agents have more space to roam around the map compared to games with 5 or 6 agents. Similarly, when investigating the effects of the map sizes, we observe that the episode length of 5 agents on larger map sizes is longer than that of smaller map sizes.

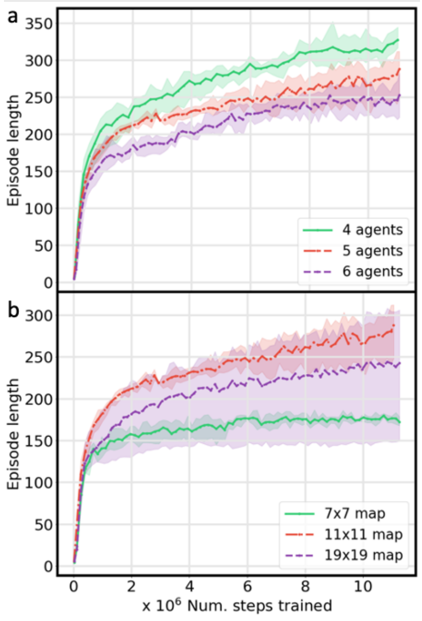

*Figure 5.* Episode lengths with (a) varying the number of agents on a $11 \times 11$ map and (b) varying the map size with five agents.

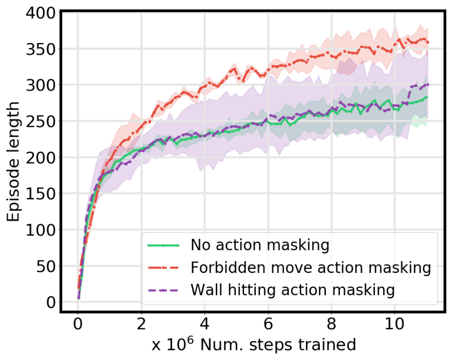

*Figure 6.* Experiments with action masking for rule 1 and 2 with 5 agents on a $11 \times 11$ map

### 5.1. Human-in-the-loop during training

**In-training action masking:** Our results as presented in Figure 6 show that the agents with in-training action masking outperforms the one without it. Action masking with rule 2 (forbidden moves) has the best performance, reaching an episode length of more than 350 at 10M steps of training. Action masking with rule 1 (wall hitting) achieves an episode length of about 300 at 10M steps whereas the agent with no heuristics achieves a bit more than 250. This verifies that including action masking improves the sample efficiency and thus accelerate the policy training.

**Reward manipulation:** We present the frequency of forbidden moves (rule 2) in Figure 7. At the beginning of training, the agents perform an average of 10 forbidden moves, which is the leading cause of death for the agents. After training for around 1M time steps, the mean frequency of forbidden moves drops to around 3. We can observe a slight reduction of the frequency of forbidden moves when comparing between the agents with and without no reward manipulation. However, as evident from the figure, the agents also manage to learn forbidden moves avoidance even without reward manipulation.

In Figure 8, we can see that the episode length for agents with in-training action masking outperform agents with reward manipulation and no heuristics. This observation is consistent with Figure 6 where forbidden move action masking improved policy training.

### 5.2. Arena testing

Table 2 shows the performances of the four agents in the Battlesnake arena to address rule 2. The four agents correspond

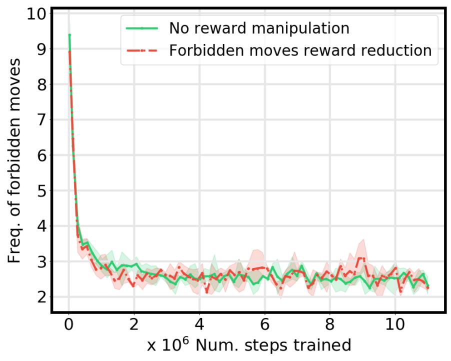

*Figure 7.* Experiments with reward manipulation for rule 2, forbidden moves with 5 agents on a $11 \times 11$ map.

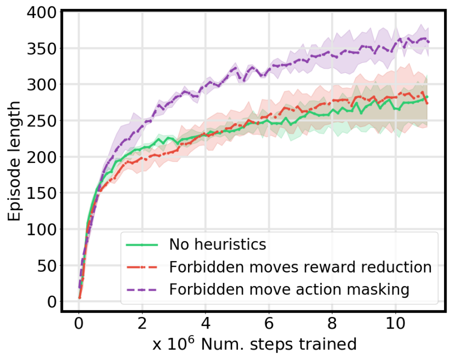

*Figure 8.* Comparison between the performance of in-training action masking, reward manipulation, and no HILL with 5 agents on a $11 \times 11$ map.

to in-training action masking, ad-hoc action overwriting, reward manipulation, and vanilla training with no heuristics. Each agent is trained on a $11 \times 11$ map for 2500 episodes. The same agents are used to tested in 1 vs. 1 competition with results presented in Table 3. We observe a consistent performance from the two tables. As expected, no HILL has the worst performance in the arena. With mean scores of 2.533 and 2.333 for in-training action masking and ad-hoc action overwriting respectively, performance of the former is slightly higher. This aligns with the trend shown in Figure 6. It is interesting to note that the agent with reward manipulation has the best performance in the arena, suggesting possible sub-optimal actions introduced during action overwriting.

## 6. Conclusion

We introduced the Battlesnake Challenge, a framework to effectively experiment and evaluate multi-agent reinforcement learning with human-in-the-loop. We formulated Battlesnake into a multi-agent RL problem and identified a set of heuristics-based rules to facilitate standardised human-in-the-loop RL research. We presented the performance of three heuristics incorporation methods. Our results suggested that the effectiveness of these methods are different during offline training and online inference. Overall, agents with HILL perform better than agents without HILL. During training, action masking improves the agents' survivability, leading to increased episode lengths. In contrast, in the Battlesnake arena, the agent with reward manipulation outperform the agent with in-training action masking. All of the code is readily available at our git repository. We look forward to contributions from the researchers and developers to build new RL based Battlesnakes.

| HILL type | Arena score | %Forbidden moves |
|---|---|---|
| No HILL | $2.200 \pm 0.846$ | 26.6% |
| In-training AM | $2.533 \pm 1.074$ | 0% |
| RM | $2.900 \pm 1.296$ | 13.3% |
| Ad-hoc AO | $2.333 \pm 1.154$ | 0 % |

*Table 2.* Scores in the Battlesnake arena and the % of deaths caused by forbidden moves. AM refers to action masking, RM refers to reward manipulation, and AO refers to action overwriting.

| | No HILL | IT AM | RM | AH AO |
|---|---|---|---|---|
| No HILL | - | 4 | 3 | 6 |
| IT AM | 6 | - | 1 | 2 |
| RM | 7 | 9 | - | 1 |
| AH AO | 4 | 8 | 9 | - |

*Table 3.* Scores (with respect to the rows) in the Battlesnake arena in a 1 vs 1 format. IT AM, AH AO and RM refer to in-training action masking, Ad-hoc action overwriting and reward manipulation respectively.

## References

Abel, D., Salvatier, J., Stuhlmüller, A., and Evans, O. Agent-agnostic human-in-the-loop reinforcement learning. *arXiv preprint arXiv:1701.04079*, 2017.

Arakawa, R., Kobayashi, S., Unno, Y., Tsuboi, Y., and Maeda, S.-i. Dqn-tamer: Human-in-the-loop reinforcement learning with intractable feedback. *arXiv preprint arXiv:1810.11748*, 2018.

Arumugam, D., Lee, J. K., Saskin, S., and Littman, M. L. Deep reinforcement learning from policy-dependent human feedback. *arXiv preprint arXiv:1902.04257*, 2019.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Bu, L., Babu, R., De Schutter, B., et al. A comprehensive survey of multiagent reinforcement learning. *IEEE*

*Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Buşoniu, L., Babuška, R., and De Schutter, B. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pp. 183–221. Springer, 2010.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pp. 4299–4307, 2017.

Gao, C., Hernandez-Leal, P., Kartal, B., and Taylor, M. E. Skynet: A top deep rl agent in the inaugural pommerman team competition. *arXiv preprint arXiv:1905.01360*, 2019.

Griffith, S., Subramanian, K., Scholz, J., Isbell, C. L., and Thomaz, A. L. Policy shaping: Integrating human feedback with reinforcement learning. In *Advances in neural information processing systems*, pp. 2625–2633, 2013.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Hausknecht, M., Mupparaju, P., Subramanian, S., Kalyanakrishnan, S., and Stone, P. Half field offense: An environment for multiagent learning and ad hoc teamwork. In *AAMAS Adaptive Learning Agents (ALA) Workshop*. sn, 2016.

Kalyanakrishnan, S., Liu, Y., and Stone, P. Half field offense in robocup soccer: A multiagent reinforcement learning case study. In *Robot Soccer World Cup*, pp. 72–85. Springer, 2006.

Knox, W. B. and Stone, P. Tamer: Training an agent manually via evaluative reinforcement. In *2008 7th IEEE International Conference on Development and Learning*, pp. 292–297. IEEE, 2008.

Knox, W. B. and Stone, P. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pp. 9–16, 2009.

Knox, W. B. and Stone, P. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pp. 5–12. Citeseer, 2010.

Knox, W. B. and Stone, P. Reinforcement learning from simultaneous human and mdp reward. In *AAMAS*, pp. 475–482, 2012.

Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I. Rllib: Abstractions for distributed reinforcement learning. *arXiv preprint arXiv:1712.09381*, 2017.

Littman, M. L. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994*, pp. 157–163. Elsevier, 1994.

Lowe, R., Wu, Y. I., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390, 2017.

MacGlashan, J., Ho, M. K., Loftin, R., Peng, B., Wang, G., Roberts, D. L., Taylor, M. E., and Littman, M. L. Interactive learning from policy-dependent human feedback. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2285–2294. JMLR. org, 2017.

Matiisen, T., Oliver, A., Cohen, T., and Schulman, J. Teacher-student curriculum learning. *IEEE transactions on neural networks and learning systems*, 2019.

Meisheri, H., Shelke, O., Verma, R., and Khadilkar, H. Accelerating training in pommerman with imitation and reinforcement learning. *arXiv preprint arXiv:1911.04947*, 2019.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Nguyen, T. T., Nguyen, N. D., and Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 2020.

Portelas, R., Colas, C., Hofmann, K., and Oudeyer, P.-Y. Teacher algorithms for curriculum learning of deep rl in continuously parameterized environments. *arXiv preprint arXiv:1910.07224*, 2019.

Rashid, T., Samvelyan, M., De Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

Resnick, C., Eldridge, W., Ha, D., Britz, D., Foerster, J., Togelius, J., Cho, K., and Bruna, J. Pommerman: A multi-agent playground. *arXiv preprint arXiv:1809.07124*, 2018.

Russell, S. and Norvig, P. Artificial intelligence: a modern approach. 2002.

Samvelyan, M., Rashid, T., Schroeder de Witt, C., Farquhar, G., Nardelli, N., Rudner, T. G., Hung, C.-M., Torr, P. H., Foerster, J., and Whiteson, S. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Saunders, W., Sastry, G., Stuhlmueller, A., and Evans, O. Trial without error: Towards safe reinforcement learning via human intervention. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2067–2069. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

Schier, M. B. and Wüstenbecker, N. Adversarial n-player search using locality for the game of battlesnake. In Becker, M. (ed.), *SKILL 2019 - Studierendenkonferenz Informatik*, pp. 109–120, Bonn, 2019. Gesellschaft für Informatik e.V.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Stone, P., Kuhlmann, G., Taylor, M. E., and Liu, Y. Keepaway soccer: From machine learning testbed to benchmark. In *Robot Soccer World Cup*, pp. 93–105. Springer, 2005.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019.

Warnell, G., Waytowich, N., Lawhern, V., and Stone, P. Deep tamer: Interactive agent shaping in high-dimensional state spaces. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Xiao, B., Lu, Q., Ramasubramanian, B., Clark, A., Bushnell, L., and Poovendran, R. Fresh: Interactive reward shaping in high-dimensional state spaces using human feedback. *arXiv preprint arXiv:2001.06781*, 2020.

Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., and Wang, J. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*, 2018.

Ye, D., Liu, Z., Sun, M., Shi, B., Zhao, P., Wu, H., Yu, H., Yang, S., Wu, X., Guo, Q., et al. Mastering complex control in moba games with deep reinforcement learning. *arXiv preprint arXiv:1912.09729*, 2019.

Zhang, K., Yang, Z., and Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *arXiv preprint arXiv:1911.10635*, 2019a.

Zhang, R., Torabi, F., Guan, L., Ballard, D. H., and Stone, P. Leveraging human guidance for deep reinforcement learning tasks. *arXiv preprint arXiv:1909.09906*, 2019b.

Zheng, L., Yang, J., Cai, H., Zhou, M., Zhang, W., Wang, J., and Yu, Y. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.