Intervening in Co-evolution

Stefan Niculae, Alejandro Marin Parra, Daniel Paul Pena, Allen Kim, Abel John
Team *ASADA* , TA: Karl Pertsch

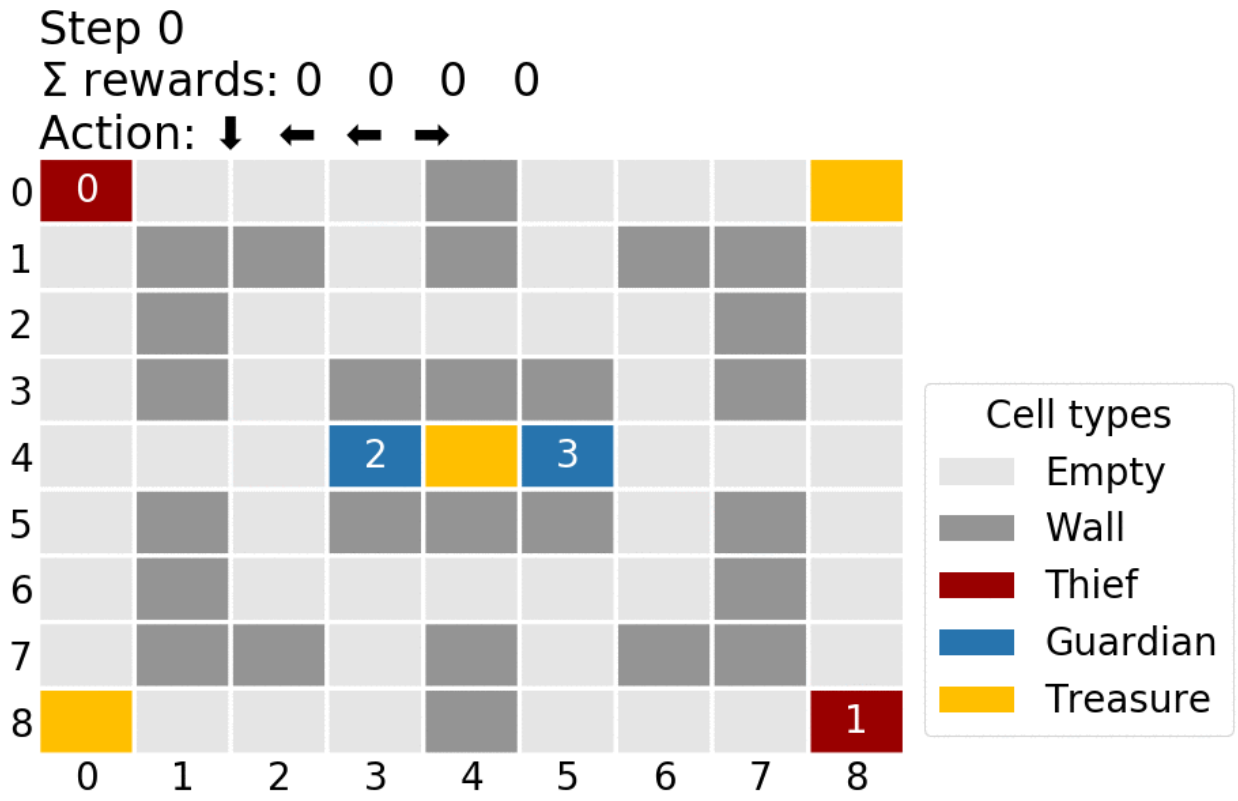
Resources:

- **Code repo**
- **Presentation**

> *Note:* gifs cannot be visualized in the PDF version of this report. Access the **online link** to see them.

Summary

We explore ways to improve Multi-Agent Reinforcement Learning (MARL) performance. We show that, in a 2D discrete zero-sum game, intervening in the co-evolution process in order to balance the relative win-rate of the two sides trains ultimately stronger strategies, for at least one side. We explore four intervention techniques: (1) helping the side that gets overpowered by providing exploration guidance, hindering the overpowering side by (2) halting their learning, (3) injecting noise in their policy and (4) constraining the quality of their latent state representation. The most well rounded improvement comes from (1) and (2), while (4) comes with a significant trade-off.

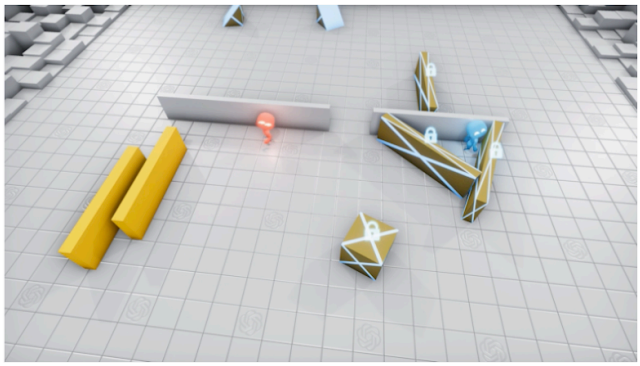


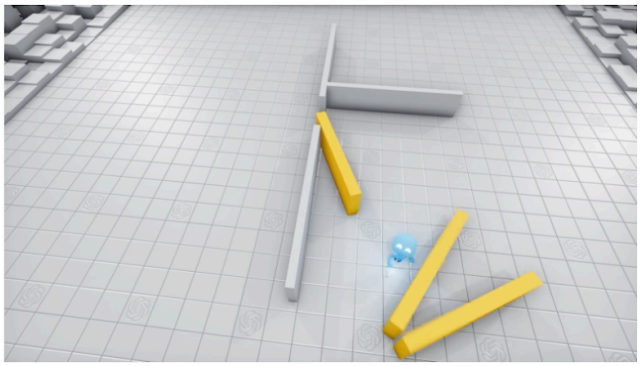
1 Introduction

Reinforcement Learning (RL) is paradigm of Machine Learning in which agents take actions in an environment in order to maximize some particular reward. In a Multi-Agent Reinforcement Learning (MARL) system, multiple agents interact with the environment and each-other, each having their own objectives.

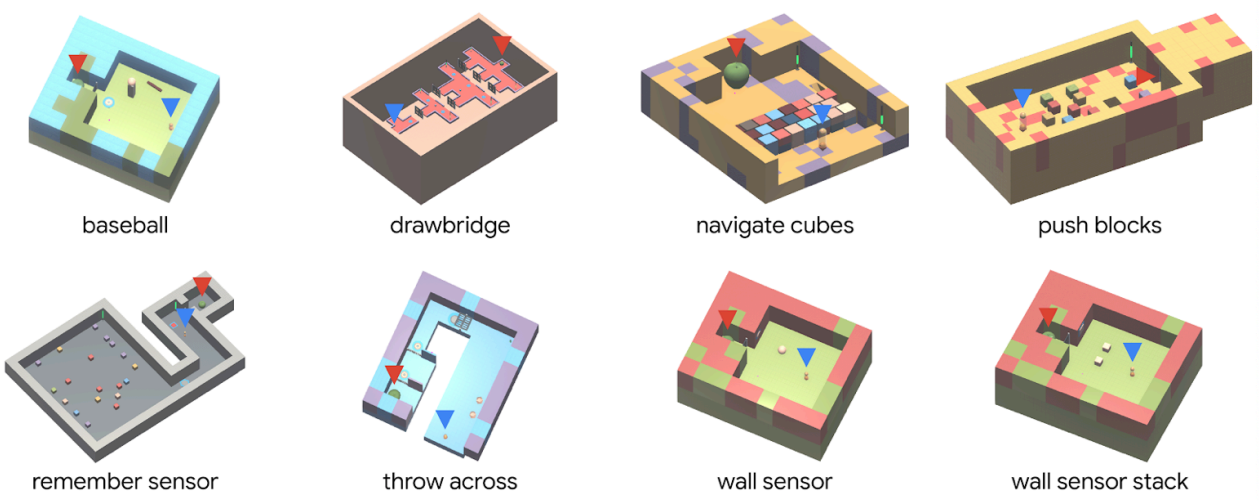
This project explores ways to better train agents in an adversarial setting, such that they both co-evolve together at a commensurate pace.

1.1 Motivation





Baker et al. [1] explore how abstract concepts such as tool use can be successfully learned through a physics-based Hide-and-Seek simulation. The hiders learn to use more and more complex strategies, such as blocking passage ways when their opponents become more and more keen at finding them. By developing stronger strategies each side facilitates the other to evolve further and develop stronger counter-strategies to match them.



RL models are notoriously hard to train and oftentimes days of training are needed to develop a strategy which is immediately apparent to humans. Paine et al. [2] showed how human demonstrations can be efficiently used (in a DQN-based algorithm) to guide the agent's exploration and point them in the right direction. Complex, sequential tasks are orders of magnitude easier to learn, in which the objectives would have otherwise not been even encountered through random, or heuristic-based exploration.

Although Hide-and-Seek environment is conceptually simple, it is computationally expensive which makes it require large models and long training times to be learned successfully. Expert demonstrations can be a way to significantly increase convergence time and success rate. But given the adversarial, multi-agent nature of the task, providing help to one team is likely to put the other at a disadvantage.

1.2 Hypotheses

In GANs, the Generator can learn to out-skill the Discriminator and fool it at all times. Because the Discriminator can no longer distinguish fake from real samples, it can provide no more information to the Generator on how to improve. Learning thus stagnates for both sides due to this unbalance in skill. On the other hand, if the Discriminator is allowed to catch on, both sides continue to grow stronger together.

We generalize this technique to a zero-sum game between two sides. Given:

- A measure of relative performance (e.g.: the win-rate between two tennis players, or how well the Discriminator can distinguish real from generated samples);
- A measure of absolute skill (e.g.: the precision, power and technique of each tennis player, or how believable generated samples are to a human eye)

We call *balanced co-evolution* a training instance in which both sides display similar relative performance throughout training.

We set out to test two hypotheses

1. Co-evolution can be balanced by intervening when relative performance becomes unbalanced;
2. Models that co-evolve in a balanced way develop ultimately better absolute skill.

2 Setup

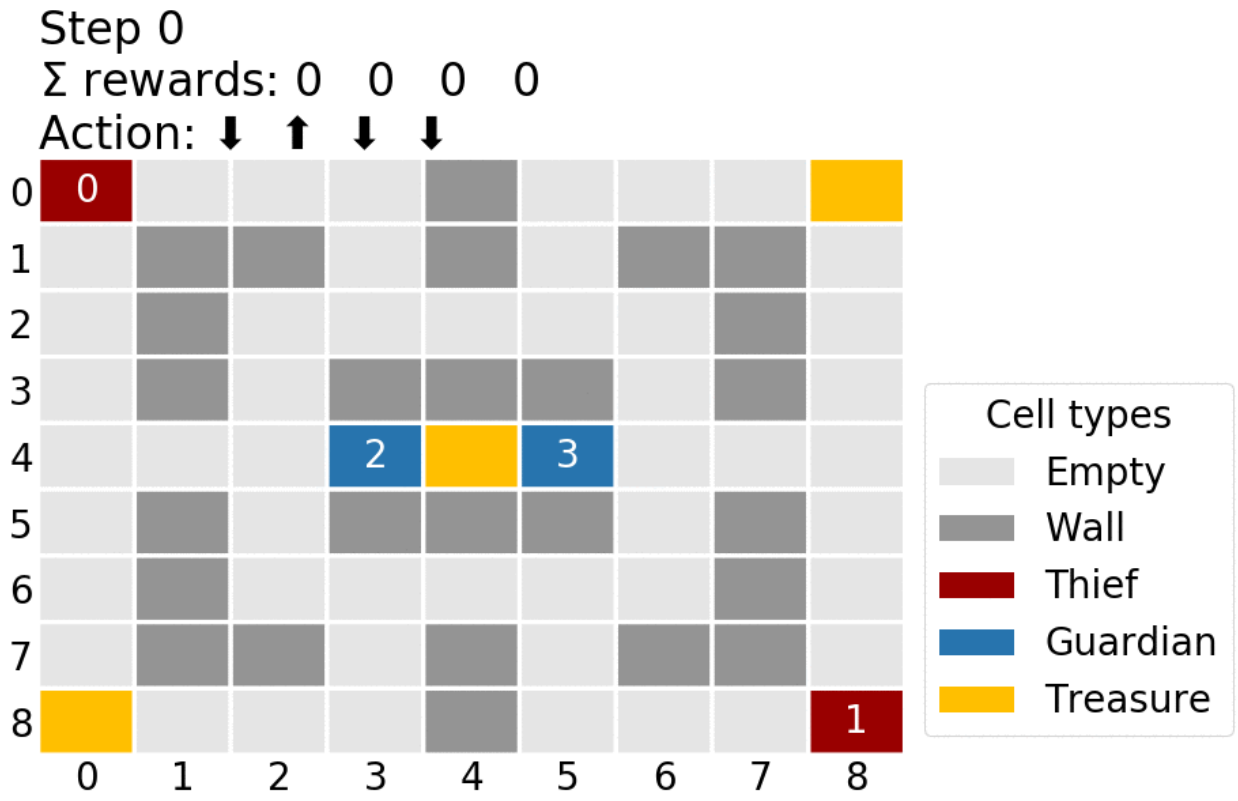
We test our hypotheses in a RL setting. This section describes the methodology: environment, agent models, evaluation considerations and intervention criteria and tactics.

2.1 Environment

The availability of research-ready single-agent RL environments is abundant, multi-agent environments are much rarer. If we bar physics-based ones due to computational constrains, the options near none. With the extra benefit of total control and a pure Python implementation, we chose to implement our own environment.

In the *Thieves-and-Guardians* environment the two teams compete in a two-dimensional discrete grid. Each team can have multiple avatars (labeled zero through the number of total avatars, for identification convenience). The *Thieves* (illustrated in red) win an episode by collecting two treasures (yellow) while the *Guardians* (blue) win by catching all thieves. An episode ends in a tie after 100 time-steps.

Agent actions Each avatar can move in four directions, one cell at a time. Trying to move to a cell occupied by a wall results in a null action, and so will trying to move to a cell occupied by a teammate. Moving to a cell occupied by an avatar of the opposing team results in the thief being caught and disappearing from the map. A thief can collect a treasure by moving on the cell occupied by it, at which point it will disappear from the map. Guardians cannot interact with treasures. At each time-step, each avatar performs one action, in order of their label.



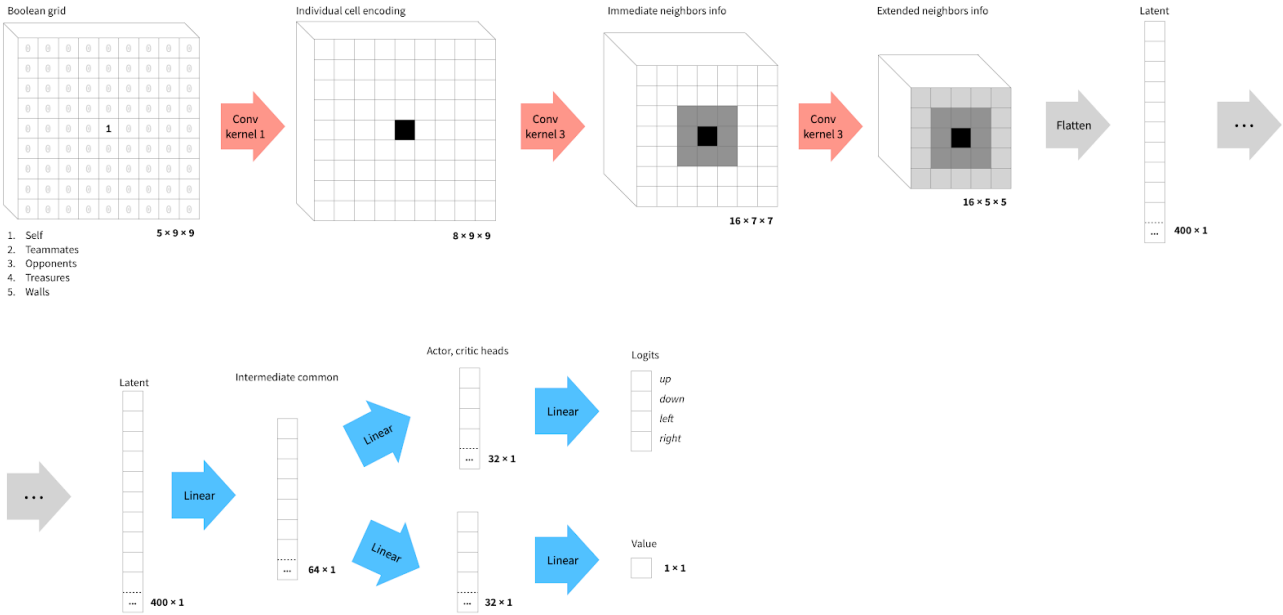
Reward function The guardian avatar who catches a thief avatar receives a reward of +1. The same reward is received by the thief avatar who collects a treasure. Due to the way actions are resolved, it is possible that a thief collects a treasure, only to be ambushed by a guardian on the other side of it, sacrificing itself after successfully collecting the treasure.

Reward engineering alternatives Policy convergence and learned strategies are heavily influenced by reward function design. We had some success with a providing a reward to thieves proportional to the number of steps they took to reach it, since the beginning of the episode. Another variation was to awarding +1 reward to all guardian avatars upon catching a thief, going by the intuition that even if one avatar was the one to catch the thief, all guardians contributed to this catch by blocking passages or otherwise forcing the thief into submission. Negative rewards, either for being caught/failing to stop a treasure collection, or a small constant for every time-step turned out to interact badly with the RL algorithm and weights optimizer, severely hindering learning. In the end, we favored parsimony and went with the formulation above.

The asymmetry of the objectives is a main cause of unbalanced co-evolution.

This environment features very high non-stationarity — every new iteration, your opponents will react differently, influenced by how you act as well.

2.2 Agents



Multi-agent action picking There are two agents, one for the thieves and one for the guardians. An agent picks an action for each avatar in the team in isolation. Each agent has their own copy of the model, to prevent intervention cross-contamination. Each avatar has their own transitions buffer, to facilitate dealing with cases in which one thief dies while the game continues.

State representation Each avatar receives a different view of environment state. The state features are made up of five 9 by 9 boolean matrices, indicating the positions of:

1. Itself
2. Teammates
3. Opponents
4. Treasures
5. Walls

The entire map is visible to all agents at all times.

Encoder architecture A 2D convolution, with 8 filters encodes each one-hot representation of cells individually (kernel size 1). Two more convolutions encode information about immediate and extended neighbors (kernel size 3). No padding is used in order to drastically reduce the dimensionality of the encoding. The result is the latent representation of the state. It is flattened and passed through a series of fully-connected layers which ultimately branch into actor and critic heads.

ReLU non-linearity and Batch Normalization is applied after each layer (except the final ones). We found that other variations of ReLU do not out-weigh their computational cost. Hyperbolic tangent non-linearities performed worse.

A recurrent component can be added to the latent representation. Despite respecting the Markovian assumption, it significantly improves training performance, but reduces computational one by an order of magnitude, rendering it unfeasible.

RL algorithm The models are trained with a Policy Gradient algorithm, with a discount factor of 0.97. PPO, and SAC (for discrete actions [10]) performed slightly, respectively significantly worse given the same training time. We suspect this to have been the case because the additional parameters to learn introduce an overhead which brings no additional benefit in the relatively short training sessions.

Inferior alternatives A (normalized) coordinates-based state representation (in which convolutional layers are replaced with fully-connected ones) failed to converge to any meaningful policies.

Deeper and/or wider (larger layer sizes), even though they have more expressive power, would have required much longer training which rendered them impractical for our constrains.

2.3 Intervention

We use the relative win-rate of the previous policy iteration (around a hundred episodes) to gauge the relative performance of thieves and agents. If the win-rate is not within a target range, for the next iteration one of four intervention tactics will be applied.

Helping the loser When the win-rate dips too low, we provide some exploration guidance to the losing side. We observed that in this case, the team's policy degenerates into a sequence of null actions, likely due to the fact that they never managed to discover sources of positive rewards. In this case, providing them with a successful trajectory would increase the chances of that the agent explores and exploits similar trajectories in the future. At each time-step, x% of the time, we force the agent to take a step in the direction of their objective (either a treasure or a goal), instead of sampling from their regular action distribution. Well trained models out-maneuver this fixed strategy, but its purpose is not to be optimal, rather help in cases where one side is

severely lacking.

Hindering the winner In contrast, when one side wins too often we want to impede their evolution, in order for the loser to catch up. We address this by either:

1. Halting the winner's training, allowing the loser to continue learning and giving them time to study their opponents better;
2. Constraining the winner's quality of the latent representation, as a non-intrusive way of regularizing the its prowess;
3. Degrading the policy by making the winner pick actions uniformly x% of the time rather than their learned policy. This is equivalent to ϵ-greedy exploration which can backfire as a technique to help the winner in the long term.

Measuring latent representation quality Constraining the latent representation is a deceivingly complex operation. Since it is nearly entirely dependent on the input features (bar the bias terms), ensuring the latent representation does not encode too much information about the raw features would be a good formulation. Due to their significantly different structures, it is extremely non-trivial to measure non-linear correlations between two arbitrary sets of values, of different sizes, with no particular prior distribution in which most of the information is encoded by the number positions rather their magnitudes [?, ?]. To address this, we take a slide a gaussian kernel across all values, to produce a differentiable soft-histogram which can be discretized. We are looking for a measure similar to Mutual Information, which is formulated as follows for discrete distribution PMFs:

$$\mathrm{I}(X;Y) = \sum_{y}\sum_{x} p_{X,Y}(x,y)\log\frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)}$$

We can have $p_X(x)$ and $p_Y(y)$ straight from the PMF, but the joint distribution makes little sense since there is no immediate pairing between value buckets of the raw and latent state representation. We then turn our attention to a related measure, cross entropy, which is formulated as follows for discrete distribution PMFs:

$$\mathrm{H}(p,q) = -\sum_{x} p(x)\log q(x)$$

We purposefully forgo the assumption of the same support, by discretizing the PMF of inputs and latent values between their minimum and maximum values. This quantity serves as an indication of how good of an encoding q is for events encoded by p. Events in this case are soft-histogram magnitudes. Since our hand-crafted encoding is obviously not optimal, encoding the input as it is also has a cost itself, so we normalize by subtracting $\mathrm{H}(p,p)$, which is precisely its entropy. Thus, the latent loss is formulated as:

$$\mathcal{L}_{\text{latent}} = \mathrm{H}(i,l) - \mathrm{H}(i,i)$$

where i and l represent the discretized soft-histograms of the input and latent features, respectively.

This loss is added to the total loss to be optimized, modulated by a coefficient to indicate its strength, and how much the encoder quality should be constrained.

Variational encoder A more theoretically-sound measure could be formulated if the latent representation was variational, representing multi-gaussian means and standard deviations. Then a straightforward Conditional VAE-like loss could be employed to constrain it to a uniform distribution. But this formulation requires a major change in model architecture, by sampling according to encoder (convolutional layers) outputs and constraining the latent distribution to be uniform. This turned out to be a too high of a cost, as such a variational architecture failed to converge on any meaningful strategy.

Input/parameter noise We do not explore adding (zero-mean gaussian) noise to the input features or model parameters, in order to avoid the trap of the opponent learning to rely on these artificial weaknesses which would be taken away as soon as they learn how to exploit them successfully.

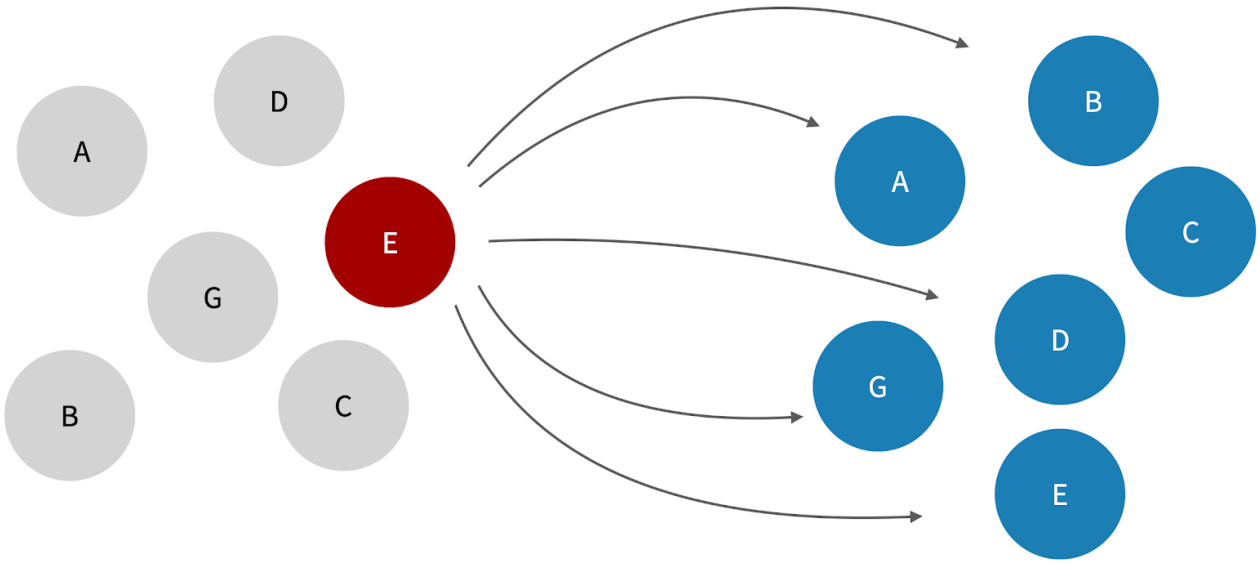
2.4 Evaluation

In order to tell whether our proposed intervention techniques improved learning, we need a way to measure the quality of learned policies, i.e.: a measure of absolute skill.

Unlike traditional RL environments, no single episode metric can tell how good a policy is. A high reward can come from playing against weak opponents. A low reward for both can mean that either both sides behave randomly without reaching their objectives, or that both are very skillfully avoiding each-other and displaying cautious behavior. The win-rate against your training opponent only shows how good you are at countering their strategy.

Match-Making Rating (MMR) schemes are not a fit either since Thief A can not directly play against Thief B. They can play the against same opponent, but again, the result only tells how good each of them is at countering that particular opponent's strategy.

The duality of the issue expresses a poses rock-paper-scissors problem. Thief A can do better than Thief B against Guardian X, but worse on Guardian Y. And since there is no direct way to measure the strength of the two Guardian teams, there is no way of translating these results into strength measurements of the two Thief teams.



League A policy is strong if it can beat a variety of opponents. Ideally we would pit it against all possible strategies and see how well it does against them. Practically, we form a *League* of opponents and use the aggregate performance against all of them as a proxy of absolute skill. A thief team plays against all guardian teams for multiple episodes and afterwards their performance is compared to those of all other thief teams. The performance against every opponent is weighed equally. The Wilcoxon test can be employed to validate the statistical difference between these paired measures without a normality assumption.

Relative metric The win-rate seems to be an obvious choice, but even barring tie handling issues, the total episode rewards can be a more expressive indication of performance. If Thief A and Thief B both get a 20% win-rate against Guardian X, but Thief A gets an average reward of 0.9 (reaches about one treasure every episode) while Thief B, 1.3 (reaches more than one treasure each episode), then Thief B is fared better against Guardian X. The relative rewards and win-rates can disagree in the case by favoring more "suicidal" strategies, in which Thieves go straight for one treasure with no intention of ever reaching the second, and disadvantaging more "cautious" strategies which go for a win instead (both treasures collected).

Aggregation Different behaviors can be encouraged by using different aggregations of results across multiple episodes against the same opponent, and afterwards the aggregation of results different across multiple opponents. Robustness can be prioritized by looking at the average result, riskier behaviors, by looking at the maximum (or a high quantile) result, and consistency can be encouraged by aiming for a low variability among match results. We chose to compare based on the average results as they offer an overall indication of all results.

Team focus When asked to report a single number for performance of a training pair (Thief and Guardian teams), needed, for example, for automatic hyper-parameter tuning, there is a choice to be made. If we care that both teams achieve good performance, the sum of their two league win-rates should be reported. In this case, it is possible that neither team reaches their absolute performance. If we only care about breeding the strongest Thief team, only its league win-rate should be reported. In this case it is likely that the ignored team will lack behind.

3 Results

In this section we compare the results of applying four intervention tactics:

1. *Latent* : add $\mathcal{L}_{\text{latent}}$ to the winner's loss;
2. *Guide* : show the loser expert demonstrations 50% of the time;
3. *Freeze* : set the winner's learning rate to zero;
4. *Noise* : have the winner sample randomly 25% of the time.

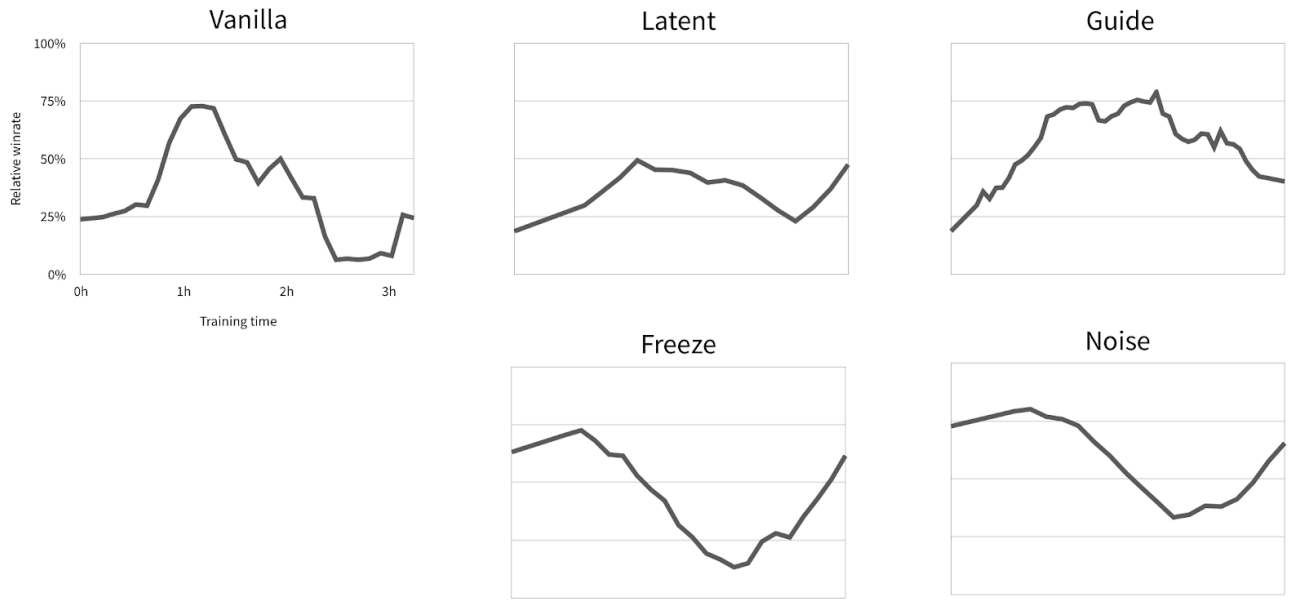
Each of these intervention tactics was applied upon when the win-rate diverged past 75%.

In addition, *Vanilla* represents no intervention.

The graphs show the best models after obtained by a non-exhaustive hyper-parameter search. All five variants were trained in the same conditions for roughly the same amount of time. Some tactics, due to being computationally more expensive were able to finish fewer policy iteration steps compared to others.

3.1 Balancing relative performance throughout training

Each of the four intervention techniques achieved more balanced training compared to Vanilla:



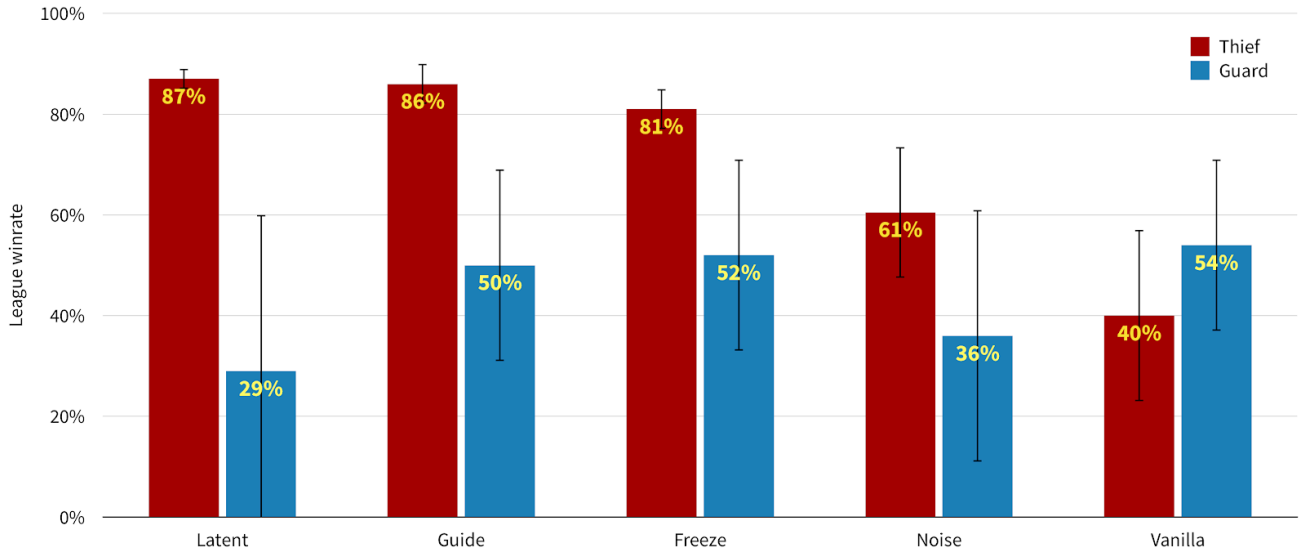
The charts show the percentage of episodes the Thief team won against their training Guardian opponents, averaged over 7 policy iteration steps.

Vanilla exhibits the highest unbalance, with a range of almost 75%. The four intervention tactics exhibit lower win-rate oscillation range, as well as having a final win-rate closer to the hypothesized ideal of 50%.

The Latent tactic shows the least oscillation which could be caused by the fact that it is a "smoother" tactic compared to the others. Rather than its effect being turned completely on or completely loss, it impacts the winner's policy in a continuous manner, by applying gradients which minimize $\mathcal{L}_{\text{latent}}$.

3.2 Improving final League performance

Each of the four intervention techniques trained a stronger thief team compared to Vanilla. Conversely, the strongest guardian team was trained using the Vanilla algorithm.



The chart above shows the performance of the best thief and the best guardian model (different seeds in each case) against a league of 48 opponents (multiple runs and intermediate checkpoints of the five variations).

The Latent thief obtained the strongest performance, very closely followed by the Guide thief and a little less closely followed by the Freeze thief. Their performance across the league is also much more consistent (error bars represent standard deviation across match results) compared to the Noise and Vanilla thieves.

The improvement in thief performance by the Latent algorithm comes at stark contrast with the guardian, which obtained much worse performance. This could be indicative of the fact that Latent guardians were good teachers for the Latent thieves, but they did not form too strong of a strategy against the League. This is further reinforced by the relative win-rate curve shown in the previous subsection. The Latent thieves never achieve over 50% win-rate (on average), yet in the League they obtain 87% win-rate, while the guardians, 29%.

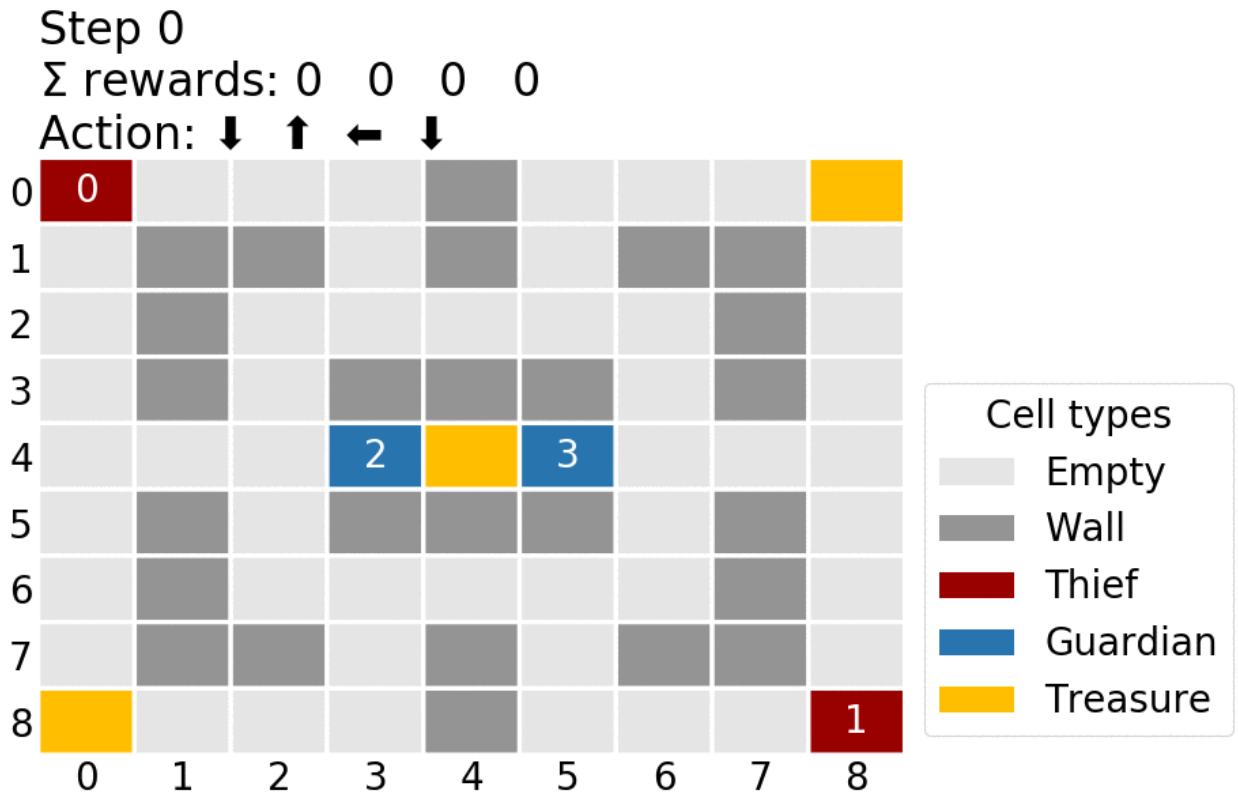
The pervasive nature of the Latent technique effect is a double-edged sword. While it continues to help the loser, it continues to hinder the winner even when $\mathcal{L}_{\text{latent}}$ is no longer explicitly optimized for. This can also be an explanation of the high drop in guardian performance on compared to Guide and Freeze is likely due to the. Constraining the latent representation has a lasting effect, while freezing the learning rate or adding noise to the policy is reversed after a while.

A reason for which the Guide or Noise techniques degraded guardian performance is that when applying them to the winner side, the lower abruptly gets exposed to a vastly different policy, which it has to fit, as its old strategy may not apply on it.

3.3 Qualitative results

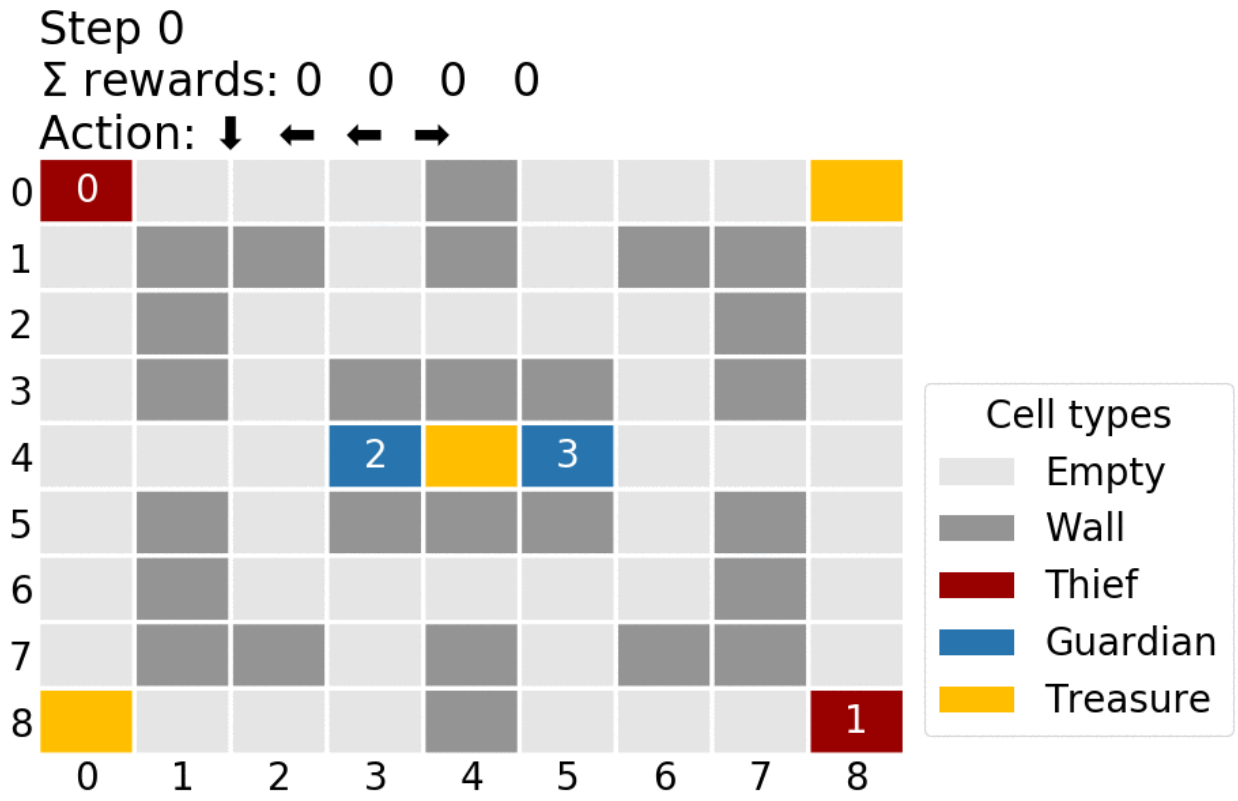
We provide a few full episode rollouts for various model pairs, to get a better sense of what behaviors agents are exhibiting.

Vanilla vs Vanilla:



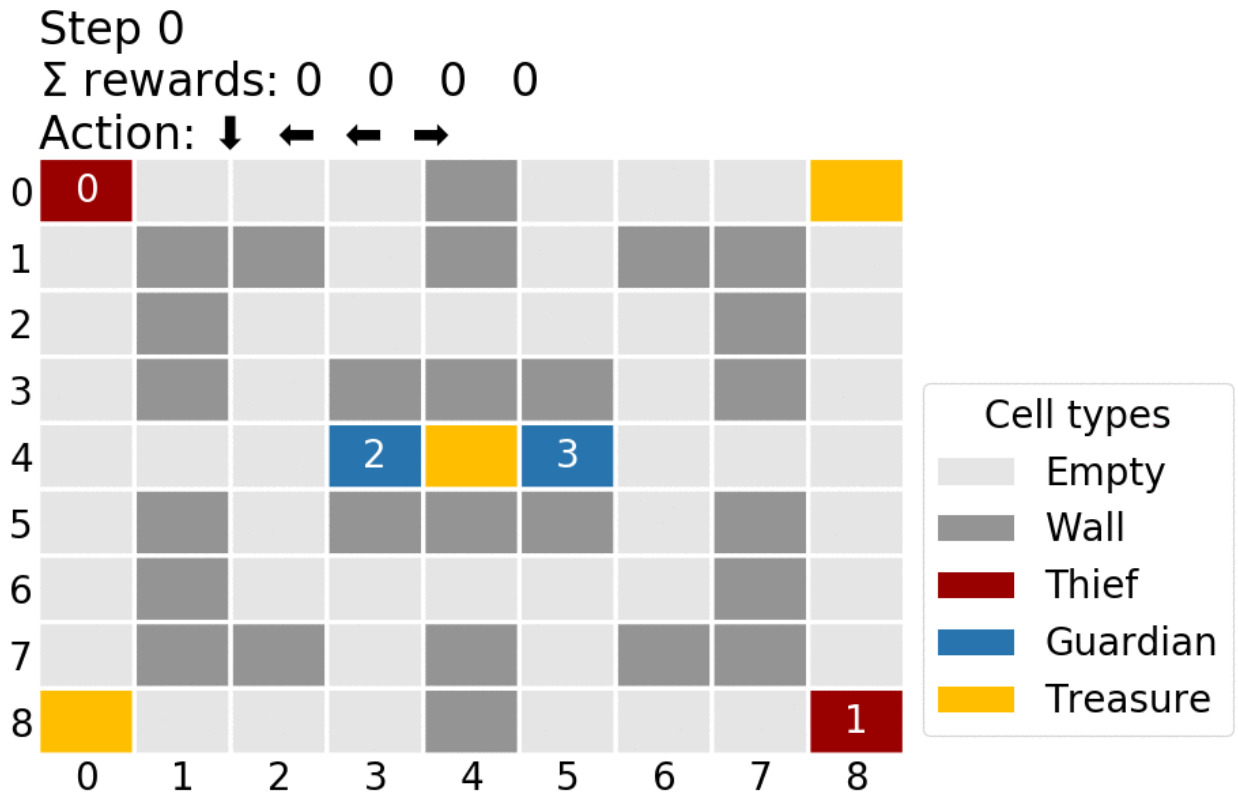
At step 3, G2 (the Guardian avatar labeled 2) blocks the access of T0 to the treasure at (0, 8) treasure then lures it towards the central one. At step 36, the two guardians manage to corner the last remaining thief.

Best thief team (Latent) vs best guardian team (Vanilla):



In the first few steps, G3 guesses T1's trajectory wrongly. Starting at step 28, the guardians, who have not fit these particular thief team's strategy enough are too slow to catch T0. Room for refinement on the thief's policy can be observed at step 24.

Failing to generalize a specialized strategy:



Even though this guardian team learned how to counter their training opponents' strategy very well, their strategy fails to generalize to unseen opponents. At step 9, the guardians block two paths to the treasure, but the thieves just go on alternate routes.

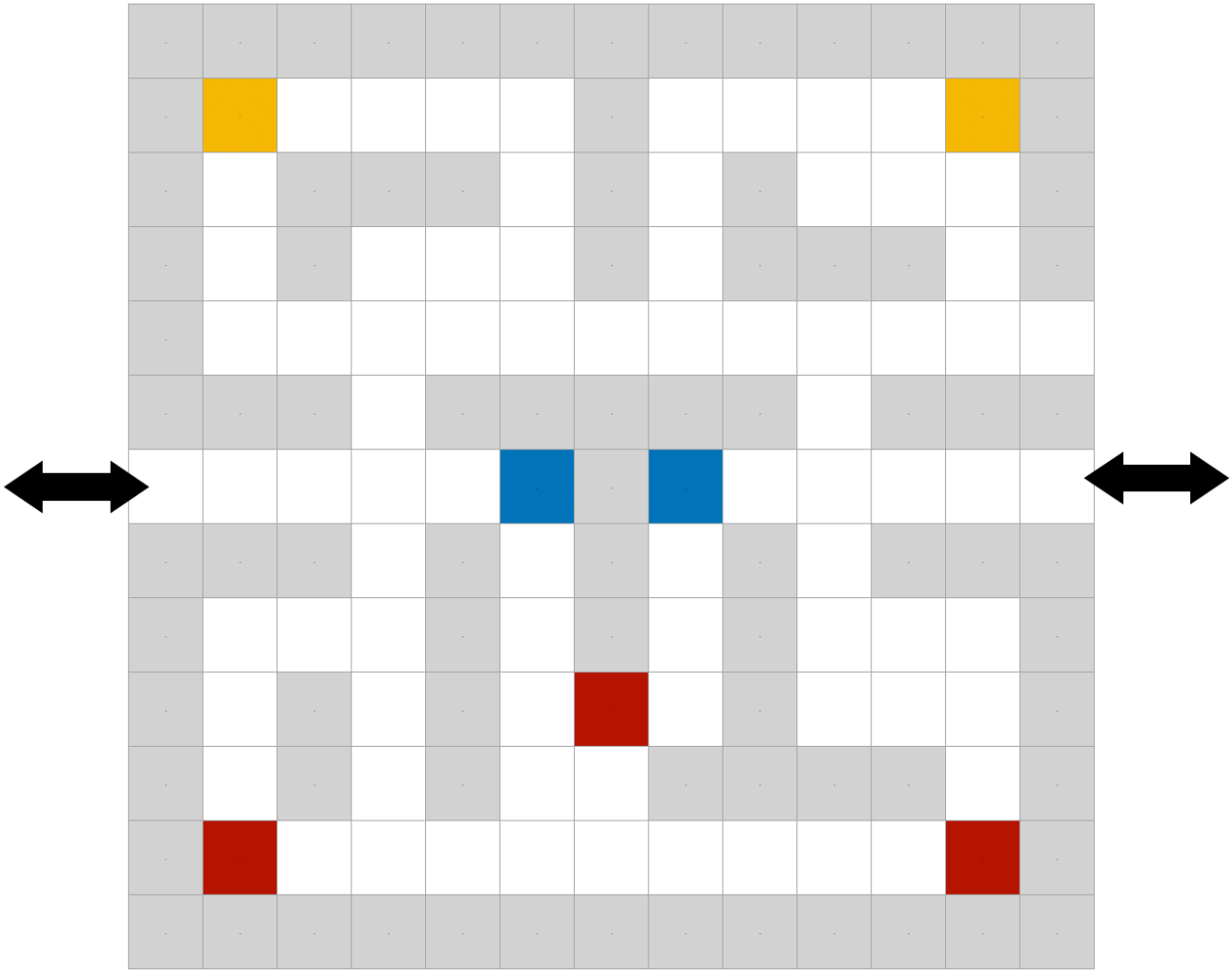
4 Future Work

Hyper-parameters Due to the computational difficulty of training the models, an in-depth tuning of hyper-parameters has not been explored. Changes in the reward function design, model hyper-parameters and RL algorithm, intervention criteria and parameters (e.g.: amount of guidance/policy noise, or latent loss coefficient) result in great differences of the obtained policies. A different architecture for policy Adding a form of communication between teammates [11]

Intervention We identify five main areas of further investigation:

1. Each of the four intervention tactics has been applied by itself, a combination of them could yield better results;
2. Other intervention criteria, such as the relative gradient magnitudes or league performances of the two teams can be used instead, or in addition to relative win-rate as a measure of relative performance;
3. Instead of providing hard thresholds for applying the intervention tactics, proportional measures can be applied instead (e.g.: reduce the learning rate of the winner the closer it gets to 100% win-rate);
4. Improving the formulation of the latent representation quality constrain[3] or employing better assimilation of guidance demonstrations[4];
5. Reverting the overpowering team to a previous checkpoint if any of the intervention tactics fail to balance

learning, or as a soft alternative, when the winner picks an action, sample x% of the time from a previous checkpoint whose skill is commensurate to their opponent.



Environment(s) Larger/more intricate scenarios that facilitate more complex strategies can better showcase intervention impact. Some ways to achieve this can be by allowing diagonal movement, or moving across multiple cells each time-step; or by creating non-euclidean topologies through screen wrap-around or portals. The utility of intervention can be validated by testing on other multi-agent environments as well, such as Sumo or Soccer, available as physics-based simulations.

Learned representations Besides measuring the quality of the learned policy, it would be interesting to explore the transferability of the built knowledge. Does it scale better (train on a small grid, evaluate on a large one); does it show stronger composability (train on individual tasks such as just avoiding a guardian or just collecting the treasure and evaluate on a scenario involving both at the same time)? The input-format variability can be bypassed by a Transformer-based encoder.

It can be more descriptive to train a World Model [12] on episodes of balanced opponents. Auto-curricula [5] can be optimized for by forcing one team to be a good teacher, rather than achieve their best.

5 Conclusions

- The proposed intervention tactics greatly improved the performance for one of the teams, while slightly degrading it for the other.
- Evaluating the performance of multiple adversarial agents with asymmetric objectives is another hurdle to the exponential complexity of training them.
- More investigation is needed, as the system exhibits extreme variance w.r.t. changes in the large amount of configuration parameters.

References

1. Baker et al — Emergent Tool Use From Multi-agent Autocurricula (2019)
2. Paine et al — Making Efficient Use of Demonstrations to Solve Hard Exploration Problems (2019)
3. Hjelm et al — Learning Deep Representations By Mutual Information Estimation And Maximization (2018)
4. Ho & Ermon — Generative Adversarial Imitation Learning (2016)
5. Sukhbaatar et al — Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play (2018)
6. Peng et al — Variational Discriminator Bottleneck: Improving Imitation Learning, Inverse RL, and GANs by Constraining Information Flow (2018)
7. Goyal et al — InfoBot: Transfer And Exploration Via The Information Bottleneck (2019)
8. Kim et al — EMI: Exploration with Mutual Information (2019)
9. Grau-Moya et al — Soft Q-learning With Mutual-information Regularization (2019)
10. Christodoulou — Soft Actor-Critic For Discrete Action Settings (2019)
11. Jaques et al — Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning (2019)
12. Ha & Schmidhuber — World Models (2018)

Appendix

A1 Parameters

- Training

Parameter	Value
Buffer size	4000
Batch size	512
Number of epochs	8
Start learning rate	0.01
End learning rate	0.001
Start entropy coefficient	0.005
End entropy coefficient	0.001
Training algorithm	PG
Number of iterations	4000
Discount	0.98
Optimizer	Adam
Adam epsilon	1e-5
Max gradient norm	5

- Environment

Parameter	Value
Maximum number of steps	100
Treasure reward	1
Guardian catches thief reward	1
Number of thieves	2
Number of guardians	2

- Encoder Architecture

Parameter	Value
Type	Convolution
Number of layers	3
Conv1 kernel size	1
Conv1 outputs	8
Conv2&3 kernel size	3
Conv2&3 outputs	32
Activation function	ReLU

- Decoder Architecture

Parameter	Value
Type	FC
FC1 size	64
FC2 size	32
Activation function	ReLU

A2 Training diagnostics

Following is an example used for checking the gradients health: