A Appendix

The appendix includes extra information of AI-Olympics benchmark environments, differences between active inference and RL, training experiments, and the Jidi online competition.

A.1 The AI-Olympics mechanics

AI-Olympics discretizes the dynamics and assumes that the acceleration between discrete time-step $t_0$ and $t_1$ is constant. This means that smaller the time interval $t_1 - t_0$ is, more accurate dynamics the simulator recovers.

A.1.1 Distance, velocity and acceleration

As illustrated in Figure 3a, at $t_0$ the agent is located at $s_0 = [s_{0x}, s_{0y}]^T$ with velocity denoted as $v_0 = [v_{0x}, v_{0y}]^T = [|v_0| \cos \theta, |v_0| \sin \theta]^T$. When a driving force $F$ and resistance $f = c|v|$ applies at $t_0$, the update at $t_1$ can be summarised as:

\[
s_1 = s_0 + v_0 \Delta t \\
a = (F - f)/m \\
v_1 = v_0 + a \Delta t = v_0 + \frac{F - f}{m} \Delta t \\
= (1 - \frac{c\Delta t}{m})v_0 + \frac{F}{m} \Delta t \\
= \gamma v_0 + \frac{F}{m} \Delta t
\]  

(3)

where the hyperparameter $\gamma$ can be seen as the decaying coefficient.

When the force (acceleration) is at a different direction from the current velocity, as illustrated in Figure 7. From $t_0$ to $t_1$, the moving distance $s_1$ can be denoted as:

\[
s_1 = \begin{bmatrix} s_{0x} + |v_0| \cos \theta_0 \Delta t \\ s_{0y} + |v_0| \sin \theta_0 \Delta t \end{bmatrix}, \quad v_1 = \gamma v_0
\]  

(4)

from $t_1$ to $t_2$, Euler methods first compute moving distance $s_2$ then update the velocity:

\[
s_2 = \begin{bmatrix} s_{1x} + |v_1| \cos \theta_0 \Delta t \\ s_{1y} + |v_1| \sin \theta_0 \Delta t \end{bmatrix} \quad v_2 = \begin{bmatrix} \gamma v_1 + \frac{F_1 \cos(\theta + \Delta \theta)}{m} \Delta t \\ \gamma v_1 + \frac{F_1 \sin(\theta + \Delta \theta)}{m} \Delta t \end{bmatrix}
\]

(5)

Whereas the semi-implicit Euler methods reverse the order:

\[
v_2 = \begin{bmatrix} \gamma |v_1| \cos(\theta_0 + \Delta \theta) + \frac{F_1 \cos(\theta_0 + \Delta \theta)}{m} \Delta t \\ \gamma |v_1| \sin(\theta_0 + \Delta \theta) + \frac{F_1 \sin(\theta_0 + \Delta \theta)}{m} \Delta t \end{bmatrix}
\]

\[
s_2 = s_1 + v_2 \Delta t
\]

A.1.2 Collision

Collision can be summarized as collision between (1) ball and line (2) ball and arc and (3) ball and ball. Collision handling often includes continuous collision detection which computes the collision time approximately, as well as calculating collision response chronologically. In brief, the ball-line collision handling compare the agent radius with point-line distance and check whether the collision point is on the line segment. The exact collision time is computed by solving a quadratic equation; the ball-arc collision handling requires checking whether the collision occurs at inside or outside, as shown in Figure 8a; ball-ball collision (shown in Figure 8b) can also be detected by simply solving a quadratic equation where no solution implies no collision occurs.
Figure 7: Illustration of the mechanics. Agent move with constant speed from $t_0$ to $t_1$, and turn from $t_1$ to $t_2$ due to applied force $F$.

Figure 8: Collision handling.

(a) Ball collides with are from both inside and outside.

(b) Ball collides with ball between the discrete time-step.

A.2 Code structure

The code structure can be summerized as follows:

- `Env_wrapper/...`: the environment wrapper we built for Jidi online AI competition events in order to fit in the back-end evaluation framework.
- `Scenario/...`: multiple scenarios wrappers built on top of the AI-Olympics engine.
- `tools/...`: some toolkits function and color assignment setting.
- `train/...`: the implemented baseline on several scenarios, including PPO and deep active inference.
(a) We test PPO and deep active inference agent on a simple running scenarios. Agent lose speed when colliding with the wall, the grey arrow implies the direction of the final.

(b) Agent observation in RGB.

Figure 9: Scenario used in experiments.

- core.py: the AI-Olympics engine, includes the dynamics, collision handling and pixel-based observation generation.
- generator.py: reads from scenario.json and create specific scenarios accordingly.
- objects.py: class object for agents, lines, arcs, etc.
- scenario.json: settings of each scenario, this includes the object in the map, the agent attributes, etc.
- viewer.py: class object for Pygame rendering.

A.3 Licenses and documentation

The AI-Olympics environment and training experiments can be found at https://github.com/jidiai/olympics_engine and are released under the MIT License. Further licensing details and all documentation can be found at this repository.

Code will be updated and maintained by the authors and contributions from the community.

B Reinforcement Learning and Active Inference

Reinforcement learning model the agent as a reward-seeking entity that either adjust the policy or learn an indicator for potential high future benefits from the environmental reward signal or from its learnt environment model. Whereas an active inference agent seeks to maintain internally homeostatic by learning an internal model to better foresee the future and act to fulfill its prior preference. One closely related paradigm is control as inference [16] (CAI) which cast decision making an inference problem and has similar optimisation objective as active inference. One main difference is that CAI introduces an extra optimality variable whereas CAI encode value into generative model directly. Furthermore, the information gain term in active inference offer a goal-directed exploration whereas CAI has only random, entropy-maximising exploration. Philosophically, CAI see perception and control as two separate processes while active inference unifies both under the free energy principle [18].

C Experiment details

For each experiment running PPO on the AI-Olympics running scenarios (map drawn in Figure 9a), the observation at each time step is a 40 × 40 array where each entry is an integer corresponds to a particular color. we used MLP actor and critic with hidden dimension [64] and ReLU activation function, taking the flatten observation as input. The detailed hyperparameter setting can be found in Table 2.
<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PPO clipping value</td>
<td>0.2</td>
</tr>
<tr>
<td>gradient clipping value</td>
<td>0.5</td>
</tr>
<tr>
<td>PPO update epoch</td>
<td>10</td>
</tr>
<tr>
<td>buffer capacity</td>
<td>1000</td>
</tr>
<tr>
<td>batch size</td>
<td>32</td>
</tr>
<tr>
<td>gamma</td>
<td>0.99</td>
</tr>
<tr>
<td>learning rate</td>
<td>1e-4</td>
</tr>
</tbody>
</table>

Table 2: PPO hyperparameter setting.

As for the deep active inference agent, the internal model is indeed a POMDP variational autoencoder (VAE) that reconstructs the current observation from the last four observations. The encoder and decoder are both two layer MLP and we pre-train the model before training starts. The deep active inference agent also contains the transition network, the policy network, the value and target value networks. The value network is updated similar to conventional value-based RL methods which minimise the MSE loss between the predicted value from the value network and the bootstrapped value computed with current reward and the next target value. The policy network is updated similarly to policy gradient except that here the variational free energy is the term to maximise. The detailed hyperparameter is listed in Table 3.

<table>
<thead>
<tr>
<th>Hyperparameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of previous observation</td>
<td>4</td>
</tr>
<tr>
<td>VAE latent state dimension</td>
<td>32</td>
</tr>
<tr>
<td>transition net latent state dimension</td>
<td>64</td>
</tr>
<tr>
<td>policy net latent state dimension</td>
<td>64</td>
</tr>
<tr>
<td>value net latent state dimension</td>
<td>64</td>
</tr>
<tr>
<td>VAE learning rate</td>
<td>1e-5</td>
</tr>
<tr>
<td>transition net learning rate</td>
<td>1e-3</td>
</tr>
<tr>
<td>policy learning rate</td>
<td>1e-4</td>
</tr>
<tr>
<td>value net learning rate</td>
<td>1e-4</td>
</tr>
<tr>
<td>buffer capacity</td>
<td>50000</td>
</tr>
<tr>
<td>batch size</td>
<td>32</td>
</tr>
<tr>
<td>target value update interval</td>
<td>25</td>
</tr>
<tr>
<td>gamma</td>
<td>0.99</td>
</tr>
<tr>
<td>entropy coefficient</td>
<td>0.5</td>
</tr>
<tr>
<td>VAE pre-trained steps</td>
<td>5000</td>
</tr>
</tbody>
</table>

Table 3: Deep active inference hyperparameter setting.

### D Jidi Platform and online competition details

Jidi is an online competition platform that provides users with massive environments, high-quality competitions, real-time discussions and fair algorithm rankings. The platform mainly provides the following five user functions:

- **Leaderboard** provides sub-ranking and total ranking of algorithms in different environments, and displays them in real time. Users can view the dynamic ranking, agent replay and detailed information of the algorithm submitted by themselves.
- **Environment** provides different environments which users can participate in and submit the algorithm. The algorithm submissions there will participate in the daily evaluation and will be displayed in the leaderboard in real time.
- **Algorithm** provides classification of popular reinforcement learning algorithms in a diagram. The diagram can be served as a learning path of RL for users.
- **Competition** provides high-quality competitions. Users can participate in every competition to get rewards.
- **Forum** provides a real-time discussion area. Users can share experiences, and find partners.
We have held several agent competitions using AI-Olympics engine as competition environment on Jidi Platform, such as [http://www.jidiai.cn/compete_detail?compete=12](http://www.jidiai.cn/compete_detail?compete=12) and [http://www.jidiai.cn/compete_detail?compete=14](http://www.jidiai.cn/compete_detail?compete=14). The first competition is based on the running scenario and the second is based on the curling scenario. They are welcomed by the platform users and both competitions attracted hundreds of participants. Most of participants are enthusiasts of agent algorithms such as college students, engineers etc.

We are now holding more competitions based on AI-Olympics, like [http://www.jidiai.cn/compete_detail?compete=17](http://www.jidiai.cn/compete_detail?compete=17) and [http://www.jidiai.cn/compete_detail?compete=18](http://www.jidiai.cn/compete_detail?compete=18). They are both based on the integrated scenario which aims to evaluate the generalization ability in multi-task environments with different goals.