

---

# Appendix

---

## A RELATED WORKS

Three types of methods have been proposed for Parameterized Markov Decision Processes (PAMDP). The methods of the first type convert the heterogeneous action space into a homogeneous one by either discretizing continuous action space ([Massaroli et al., 2020]) or transferring discrete actions into continuous space [Hausknecht and Stone, 2016]. By doing this, conventional RL algorithms can be straightly applied in this setting. However, discretizing all dimensions of the continuous action suffers from the loss of control accuracy and the scalability issue (due to the exponentially exploring number of discretized actions); while casting all discrete actions into continuous ones enlarges the original action space, resulting in additional difficulties in approximation and generalization [Li et al., 2021].

In order to avoid the problems caused by direct homogeneous motion space. The methods of the second type directly build separate policies for the discrete actions and the continuous ones. Hybrid PPO (HPPO) [Fan et al., 2019] builds multiple network heads (one for discrete actions and the others for the continuous parameters of each discrete action) to learn the hybrid policy. A similar idea is also adopted in Action Branching PPO. However, building a separate continuous policy network for each discrete action will significantly enlarge the model size, resulting in additional difficulties in optimization. Besides, both these two types of methods neglect the natural dependence between each discrete action and the corresponding continuous parameters, thus their learning processes are inefficient especially when the hybrid action space becomes high-dimensional.

To improve the sample efficiency, The methods of the third type explicitly incorporate such dependencies into the model design. Parameterized DQN (PDQN) [Xiong et al., 2018] proposes a hybrid structure by combining DQN [Silver et al., 2016] with DDPG [Lillicrap et al., 2016], where the Q-network of DQN directly takes the actor’s outputs of DDPG (i.e., the continuous parameters for all discrete actions) as additional inputs. However, the dependence of PDQN’s Q-values on all parameters actions causes a *false gradients* issue and can lead to suboptimal action selection. To address the *false gradients* issue. *False gradients* means that due to the serial training of discrete action strategies and all continuous action strategies, the gradient of discrete action policy affects the training of unrelated continuous action modules, making the overall strategy suboptimal. Bester et al. [2019] further design a Multi-Pass Q-Network (MP-DQN), making each discrete action’s Q-value only depends on its corresponding continuous parameters. Similar ideas also have been applied to Soft Actor Critic (SAC) based methods [Delalleau et al., 2019] and PPO based methods [Ma et al., 2021]. And [Fu et al., 2019] further extends PDQN to multi-agent RL settings. Most recently, Li et al. [2021] propose Hybrid Action Representation (HyAR), which uses an embedding table and a conditional Variational Auto-Encoder (VAE) to convert both the discrete and the continuous actions into a more compact latent space. Then, the policy is trained in the latent action space via Twin Delayed Deep Deterministic policy gradient (TD3) algorithm [Fujimoto et al., 2018]. HyAR achieves state-of-the-art (SOTA) performance on typical hybrid control tasks with higher sample efficiency, especially for high-dimensional action spaces. However, the embedding table and the VAE have to be periodically re-trained with the policy updating. Thus, HyAR is unstable and difficult to train due to the co-evolution of the latent action embedding and the RL policy. **Apart from the action-dependency modeling, another challenge affecting learning efficiency is the hard exploration problem.** As far as we know, this problem has not been discussed in the previous work, and our work is devoted to solving this problem so as to further improve the learning efficiency of hybrid action space control tasks.

## B EXPERIMENTAL DETAILS

Our codes are implemented with Python 3.7 and Torch 1.7.1 experiments were run on a single NVIDIA GeForce GTX 2080Ti GPU. Each single training trial ranges from 4 hours to 10 hours, depending on the algorithms and environments.

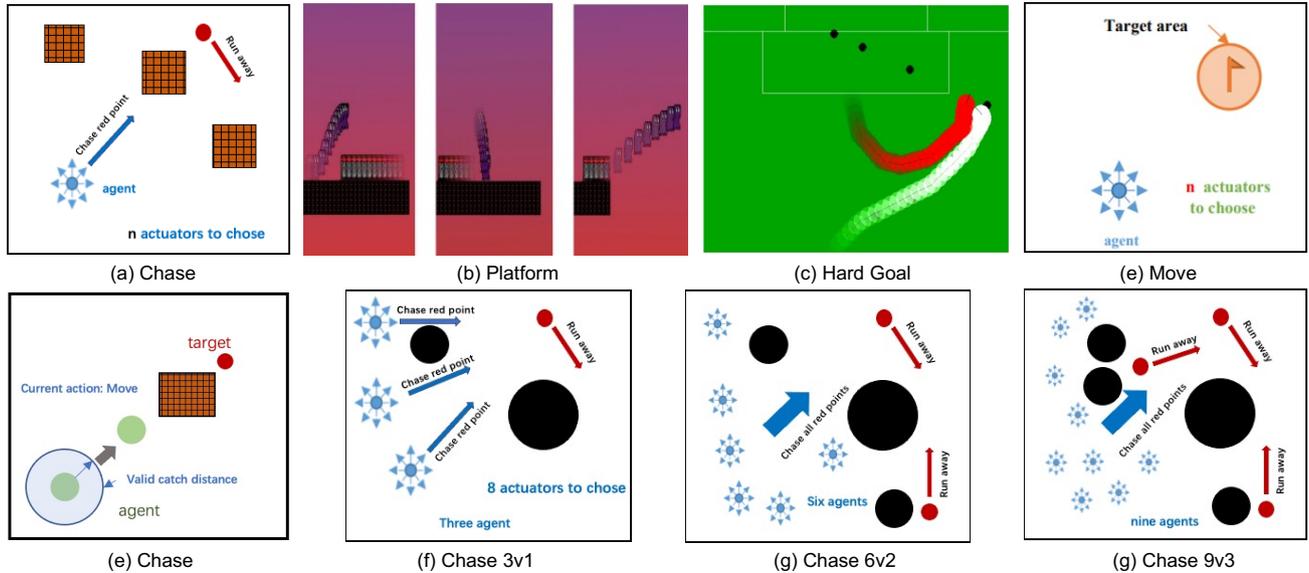


Figure 1: Benchmarks

**The multi-agent hybrid control tasks** (Figure 1 (f) 3v1,(g) 9v3, require cooperation among agents to achieve goals. The action space of the agent is the same as single agent Chase, but multiple agents need to cooperate to catch all the targets. And need to bypass roadblocks. Both environments are divided into easy and hard tasks, with the harder tasks having targets that move faster and are smaller and harder to capture.

**Single agent Environments.** We conduct our experiments on several hybrid action environments and a detailed experiment description is below.

- **Platform:** The agent needs to reach the final goal while avoiding the enemy or falling into the gap. The agent needs to select the discrete action (run, hop, leap) and determine the corresponding continuous action (horizontal displacement) simultaneously to complete the task. The horizon of an episode is 20.
- **Hard Goal:** Three types of hybrid actions are available to the agent including kick-to  $(x, y)$ , shoot-goal  $(h)$ . The shot-goal action and split into ten parameterized actions by dividing the goal line equidistantly. The continuous action parameters of each shot action will be mapped to a region in the goal line. The horizon of an episode is 50.
- **Catch Point:** The agent should catch the target point (orange) in limited opportunity (10 chances). There are two hybrid actions move and catch. Move is parameterized by a continuous action value which is a directional variable and catch is to try to catch the target point. The horizon of an episode is 20.
- **Chase (designed by us):** The agent needs to control  $n$  equally spaced actuators to reach the target area (orange). The agent can choose whether each actuator should be on or off. Thus, the size of the action set is exponential in the number of actuators that is  $2^n$ . Each actuator controls the moving distance in its own direction.  $n$  controls the scale of the action space. As  $n$  increases, the dimension of the action will increase. The horizon of an episode is 25.
- **Multi-agent Chase (designed by us):** The action space of the agent is the same as Chase, but multiple agents need to cooperate to catch all the targets. And need to bypass roadblocks.

## C MULTI-AGENT HYBRID CTRLFORMER

Hybrid CtrlFormer is an actor-critic framework and can be easily combined with existing value-based or actor-critic based multi-agent algorithms. The most straightforward way to introduce Hybrid CtrlFormer into multi-agent RL is using

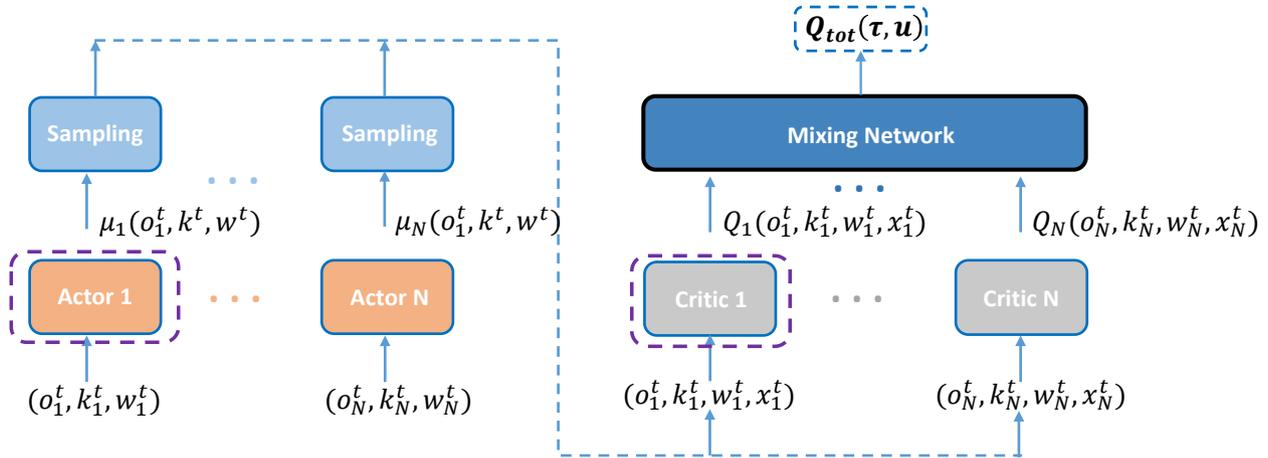


Figure 2: Multi-agent Hybrid CtrlFormer

independent Q-learning (IQL) [Tan, 1993]. However, this simple approach suffers from the non-stationarity issue caused by the changing policies of the learning agents, and thus the learning process is unstable. Recently, a great deal of work has shown that using centralised training can naturally handle the non-stationary problem [Yang et al., 2020, Wang et al., 2020]. Inspired by FACMAC [Peng et al., 2021], we utilize the value decomposition mechanism to coordinate the policy update over hybrid action spaces among agents.

The overall architecture of MA Hybrid CtrlFormer is shown in Figure 2. In MA-Hybrid CtrlFormer, each agent’s continuous policy is  $\mu_i(e(s, k, w); \theta)$  and each agent’s Q-network is  $Q_i(e(s, k, w), x_{kw}; \omega)$  and all agents share the same parameters ( $\theta$  and  $\omega$ ) to facilitate training. All agents share a centralised critic  $Q_{tot}$  which can be factored as:

$$Q_{tot}(\tau, \mathbf{u}, \mathbf{s}; \psi) = h_\psi(\mathbf{s}, \{Q_i(e(s, k, w), x_{kw}), i \in \{1, \dots, N\}\}) \quad (1)$$

where  $\tau$  denotes local action observation histories and  $\mathbf{s}$  means the global state,  $\mathbf{u}$  denotes all agents’ hybrid actions,  $\{Q_i(e(s, k, w), x_{kw}), i \in \{1, \dots, N\}\}$  are the individual Q-values of all agents ( $N$  is the agent number), and  $h_\psi$  is a non-linear monotonic function parameterized as a mixing network with parameters  $\psi$ , as in QMIX [Rashid et al., 2018]. The centralised but factored critic is trained by minimizing the following loss:

$$L(\psi; \omega) = \mathbb{E}_D [(y^{tot} - Q_{tot}(\tau, \mathbf{u}, \mathbf{s}; \psi))^2] \quad (2)$$

where  $y^{tot} = r + \gamma Q_{tot}(\tau', \mu', s', \psi^-)$ , and  $\mu'$  represents the target hybrid actions generated by all agents’ target individual Q-networks and target individual actors. The individual continuous policy is optimized according to the following loss:

$$\mathcal{L}^\mu(\theta) = -Q_{tot}(\mathbf{s}, \{Q_i(e(s, k, w), \mu(e(s, k, w); \theta); \omega), \dots\}) \quad (3)$$

## D MULTI-AGENT EXPERIMENTS

In multi-agent tasks, five SOTA approaches are selected as baselines: Independent PDQN [Xiong et al., 2018], MAPDQN [Fu et al., 2019], Independent HPPO [Fan et al., 2019], Independent HyAR [Li et al., 2021] and CTDE-HyAR [Li et al., 2021]. Hybrid CtrlFormer contains one layer transformer( with one MLP, and a unique attention mechanism). Thus the network depth of algorithms is the same. PDQN, HHQN, HyAR, and Hybrid CtrlFormer are all implemented based on the same architecture DDPG. CTDE-HyAR is based on FACMAC [Peng et al., 2021].

The experimental results are shown in Figure 3, in which the total average reward curves are smoothed for visual clarity. The results show that MA Hybrid CtrlFormer, denoted as MAHT in the figure, significantly outperforms all baselines in all six scenarios. Cooperative tasks cause the environment to be unstable [Rashid et al., 2018] (the environment can be changed by the actions of other agents), which makes the state information more complex. This phenomenon reinforces the need for exploration efficiency. Moreover, the performance of our method is stable in the large-scale agent scenarios, which proves that MA Hybrid CtrlFormer can not only realize effective exploration but also overcome the factors of environmental instability to achieve efficient coordination between agents by combining the value decomposition method.

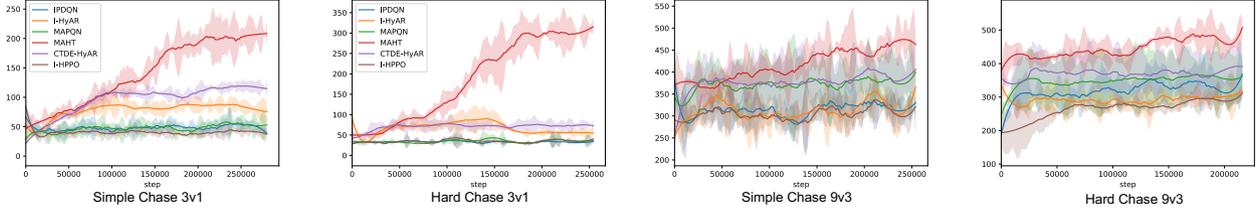


Figure 3: Comparisons of algorithms in multi-agent environments.  $y$ -axis denotes the average reward. The curve and shade denote the mean and a standard deviation over 5 runs. We abbreviate *Multi-agent Hybrid CtrlFormer* to *MAHT*.

Hyperparameter	HPPO	PADDPG	PDQN	HHQN	HyAR-DDPG	Hybrid CtrlFormer
Actor Learning Rate	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$	$1e^{-4}$
Critic Learning Rate	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$3e^{-4}$	$3e^{-4}$	$1e^{-4}$
Target Actor Update Rate	$N$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$1e^{-3}$	$5e^{-3}$
Target Critic Update Rate	$N$	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$	$1e^{-2}$	$5e^{-3}$
Batch Size	128	128	128	128	128	128
Buffer Size	$1e5$	$1e5$	$1e5$	$1e5$	$1e5$	$1e5$

Table 1: Comparisons of the baselines regarding the average win rate at the end of the training process with the corresponding standard deviation over 5 runs. Values in bold indicate the second-best results in each environment. And Red Values denote the best performances.

## E HYPERPARAMETERS

For all of our experiments, we use the raw state and reward from the environment without any normalization or scaling. Additionally, no regularization is applied to the actor and critic models in any of the algorithms. To encourage exploration, an exploration noise sampled from  $N(0, 0.1)$  is added to all baseline methods when selecting actions. In the case of Hybrid CtrlFormer, the continuous action space of each environment is uniformly divided into 8 sub-regions in sequential order. Concretely, the continuous action space of the environment is  $[-1, 1]$ , and the 8 sub-regions are defined as follows:  $[-1, -0.75)$ ,  $[-0.75, -0.5)$ ,  $[-0.5, -0.25)$ ,  $[-0.25, 0)$ ,  $[0, 0.25)$ ,  $[0.25, 0.5)$ ,  $[0.5, 0.75)$ , and  $[0.75, 1.0]$ . The  $c$  parameter of the MCTS is  $1/\sqrt{2.0}$ , which is the optimal setting that has been tested extensively and is the general default setting for MCTS. The number of single agent training episodes is 6000 to ensure that all algorithms converge. The episode number for MARL frameworks is 10000. The discount factor used is 0.99, and we employ the Adam optimizer for all algorithms. Table 1 summarizes the common hyperparameters used in all of our experiments.

## F NETWORK STRUCTURE

PADDPG and PDQN are implemented with reference to <https://github.com/cycraig/MP-DQN>. For a fair comparison, all the baseline methods have the same network structure (except for the specific components of each algorithm) as our HyAR-TD3 implementation. For PDQN, PADDPG, HPPO paper does not provide open-source code and thus we implemented it by ourselves according to the guidance provided in their paper. For HPPO, the discrete actor and continuous actor do not share parameters (better than sharing parameters in our experiments). We use a two-layer feed-forward neural network of 256 and 256 hidden units with ReLU activation (except for the output layer) for the actor network for all algorithms. For PADDPG, PDQN and HHQN, the critic denotes the Q-network. For HPPO, the critic denotes the V network. Some algorithms (PADDPG, HHQN) output two heads at the last layer of the actor network, one for discrete action and another for continuous action parameters. For HyAR-DDPG, we use the author’s open-source code. In order to prove that the Hybrid CtrlFormer framework is not redundant and will not cause training difficulties due to the division of continuous motion space, the structure details are shown in Tab 2.

Model Component	Layer(Name)	Structure
Critic	Fully Connected	( <i>state dim</i> , 128)
	Activation	ReLU
Transformer	Fully Connected	(128, 1)
	Fully Connected	( <i>state dim</i> , 128)
	Activation	ReLU
	self-attention	(128, 128), ReLU
Actor	Fully Connected	(128, <i>subregion number</i> )
	Fully Connected	(128, 1)

Table 2: Network structures for the Hybrid CtrlFormer including, the discrete action Q network, one layer transformer, and the actor-critic architecture

Benchmarks	HPPO	PADDPG	PDQN	HHQN	HyAR-DDPG	Hybrid CtrlFormer
Platform	2h4m57s	2h24m33s	2h56m06s	2h26m17s	3h11m41s	2h48m51s
Chase	20h34m08s	20h28m14s	22h06m36s	22h27m12s	23h20m06s	22h12m35s
Goal	7h12m23s	7h15m57s	7h46m05s	8h03m02s	8h53m26s	8h41m29s

Table 3: The convergence rate of Hybrid CtrlFormer is in the middle among all methods.

## G MODEL SCALE AND PERFORMANCE ANALYSIS

Despite utilizing both Transformer and MCTS, Hybrid CtrlFormer does not increase the network depth, and the computational demand remains at the same order of magnitude as previous methods. We conducted experiments to test the convergence time of all methods (with Hybrid CtrlFormer sampling 32 times per step) in multiple scenarios on a single NVIDIA GeForce GTX 2080Ti GPU (see Tab 3). The results demonstrate that the combination of multiple modules has a limited effect on the time performance of our method, and its convergence speed remains at the same order of magnitude as previous work.

### References

- Craig J. Bester, Steven D. James, and George Dimitri Konidaris. Multi-pass q-networks for deep reinforcement learning with parameterised action spaces. *CoRR*, abs/1905.04388, 2019. URL <http://dblp.uni-trier.de/db/journals/corr/corr1905.html#abs-1905-04388>.
- Olivier Delalleau, Maxim Peter, Eloi Alonso, and Adrien Logut. Discrete and continuous action representation for practical rl in video games. *CoRR*, abs/1912.11077, 2019. URL <http://dblp.uni-trier.de/db/journals/corr/corr1912.html#abs-1912-11077>.
- Zhou Fan, Rui Su, Weinan Zhang, and Yong Yu. Hybrid actor-critic reinforcement learning in parameterized action space. In Sarit Kraus, editor, *IJCAI*, pages 2279–2285. ijcai.org, 2019. URL <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2019.html#FanS0019>.
- Haotian Fu, Hongyao Tang, Jianye Hao, Zihan Lei, Yingfeng Chen, and Changjie Fan. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. In Sarit Kraus, editor, *IJCAI*, pages 2329–2335. ijcai.org, 2019. URL <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2019.html#FuTHLCF19>.
- Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.
- Matthew J. Hausknecht and Peter Stone. Deep reinforcement learning in parameterized action space. In Yoshua Bengio and Yann LeCun, editors, *ICLR (Poster)*, 2016. URL <http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#HausknechtS15a>.
- Boyan Li, Hongyao Tang, Yan Zheng, Jianye Hao, Pengyi Li, Zhen Wang, Zhaopeng Meng, and Li Wang. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. *arXiv preprint arXiv:2109.05490*, 2021.

- Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016. URL <http://dblp.uni-trier.de/db/conf/iclr/iclr2016.html#LillicrapHPHETS15>.
- Jun Ma, Shunyi Yao, Guangda Chen, Jiakai Song, and Jianmin Ji. Distributed reinforcement learning with self-play in parameterized action space. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1178–1185. IEEE, 2021.
- Stefano Massaroli, Michael Poli, Sanzhar Bakhtiyarov, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Neural ordinary differential equation value networks for parametrized action spaces. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.
- Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.
- Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 4295–4304. PMLR, 2018.
- David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016. doi: 10.1038/nature16961. URL <https://doi.org/10.1038%2Fnature16961>.
- Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.
- Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. *arXiv preprint arXiv:2008.01062*, 2020.
- Jiechao Xiong, Qing Wang, Zhuoran Yang, Peng Sun, Lei Han, Yang Zheng, Haobo Fu, Tong Zhang, Ji Liu, and Han Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.
- Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.