A Author Statement

The authors of this work would like to state that we bear full responsibility for any potential violation of rights, including copyright infringement or unauthorized use of data. We affirm our commitment to conducting this research in accordance with ethical guidelines and legal requirements.

We further guarantee that we will ensure access to the data[4] and the framework code[5] used in this study, making them available to interested researchers for verification and replication purposes. Additionally, we are committed to providing the necessary maintenance and support to ensure the longevity and accessibility of the data. For datasets, we have plans to consistently offer more datasets in the future. These datasets will include larger sizes for larger models, higher levels for expert agents, and novel design factors for other research directions. Updating our datasets is an ongoing and long-term effort, and we welcome contributions from the community. Regarding benchmarks, we will actively monitor the latest state-of-the-art (SOTA) algorithms in the offline RL domain and integrate them into our benchmarks. Additionally, we will develop new algorithms within the benchmarks based on existing datasets and baselines. This ensures that our benchmarks remain up-to-date and reflect the advancements in offline RL research.

Should any concerns or inquiries arise regarding the contents of this work or the associated data, we encourage readers and fellow researchers to contact us directly. We are dedicated to addressing any issues promptly and transparently to uphold the integrity of our research.

B Limitations and Future Works

In our future endeavors, we plan to integrate our framework with a large-scale deep reinforcement learning platform namely *KaiwuDRL*, specifically designed to support Honor of Kings. By doing so, we will gain access to greater computational resources, enabling us to delve deeper into our current research endeavors and expand our investigations.

C Datasets Details

C.1 HoK1v1

Table 3 presents the details of datasets in HoK1v1 task. The datasets are collected by recording the battle trajectories of pre-trained models as described in Appendix F. Typically, each dataset has a capacity of 1000 trajectories. The default heroes chosen for both camps are *luban*, with its Summoner Spells set to *frenzy*. However, in specific scenarios such as *Generalization* or *Multi-Task* settings, we employ a random selection of heroes from a predefined set, *multi_hero*. This set comprises five heroes, *{luban, direnjie, houyi, makeboluo, gongsunli}*, each with their Summoner Spells as *{ frenzy, frenzy, frenzy, stun, frenzy }*, respectively. The win rate of the behavior policy is recorded in the column labeled *Win_rate* for reference. The column labeled *Levels* denotes the levels of opponents used for evaluation. Generally, a level of "1" is used for datasets with the "norm" prefix, while a level of "5" is used for datasets with the "hard" prefix. This distinction indicates varying levels of difficulty.

In the *Generalization* category, "norm_general" and "hard_general," have their corresponding datasets. For example, to sample the "norm_general" dataset, we let the level-1 model fight with level-0, level-2, and level-4 models. However, during the test stage, we assess the generalization capabilities of the trained model by letting it fight against the level-1 model. Details about how we sample the *generalization* datasets can be referred to Table. 7. The latter four experiments do not require extra datasets. For example, in the "norm_hero_general" experiment, we directly use the model trained on the "norm_medium" dataset and let the model control different heroes. This is possible because the "norm_medium" dataset only contains the fixed default hero "luban." Therefore, we use the model trained on this dataset to test its generalization ability at controlling different heroes.

[4] https://sites.google.com/view/hok-offline

[5] https://github.com/tencent-ailab/hokoff

In the *Sub-Task: Destroy Turret* category presented in Table 4, there are three datasets sampled, each consisting of 100 trajectories. Notably, these datasets lack an opponent hero, making them simpler in nature. This design choice allows for broad applicability, diversity, and cost-effectiveness in research endeavors.

The primary objective in the *Sub-Task: Destroy Turret* scenarios is to efficiently dismantle the enemy's turret and crystal, with the enemy hero removed. Consequently, we adopt the number of game frames elapsed from the start of the game until the crystal's destruction as our evaluation protocol. Equation 1 outlines the scoring methodology employed, following a similar approach as presented in [6]. The score is normalized by two factors: *random_frame_length*, set to 2880, and *expert_frame_length*, set to 1812. A higher score is achieved by minimizing the time required to destroy the crystal.

In addition, we have generated violin charts to represent the distribution of episode returns in each dataset as shown in Fig. 3. We calculate episode returns using the formula $R = \sum_{t=0}^{T} \gamma^t r^t$, where gamma is set to 1.0 to showcase the overall rewards obtained throughout an entire episode. For the *Sub-Task: Destroy Turret* datasets of HoK1v1, we have normalized the scores based on Equation 1. The violin charts demonstrate the diverse distribution of episode returns within our datasets.

Table 3: Details of datasets in HoK1v1 game mode

Factors	Datasets/Experiments	Capacity	Heroes	Oppo_heroes	Win_rate	Levels
Multi-Difficulty	**norm_poor**	1000	default	default	12%	1
	norm_medium	1000	default	default	50%	1
	norm_expert	1000	default	default	88%	1
	norm_mixed	1000	default	default	50%	1
	hard_poor	1000	default	default	6%	5
	hard_medium	1000	default	default	50%	5
	hard_expert	1000	default	default	84%	5
	hard_mixed	1000	default	default	45%	5
Generalization	**hard_general**	1000	default	default	90%	5
	norm_general	1000	default	default	46%	1
	norm_hero_general	-	multi_hero	default	-	1
	hard_hero_general	-	multi_hero	default	-	5
	norm_oppo_general	-	default	multi_hero	-	1
	hard_oppo_general	-	default	multi_hero	-	5
Multi-Task	**norm_multi_level**	1000	default	default	50%	1
	hard_multi_level	1000	default	default	50%	5
	norm_multi_hero	1000	multi_hero	default	23%	1
	norm_multi_oppo	1000	default	multi_hero	77%	1
	norm_multi_hero_oppo	1000	multi_hero	multi_hero	50%	1

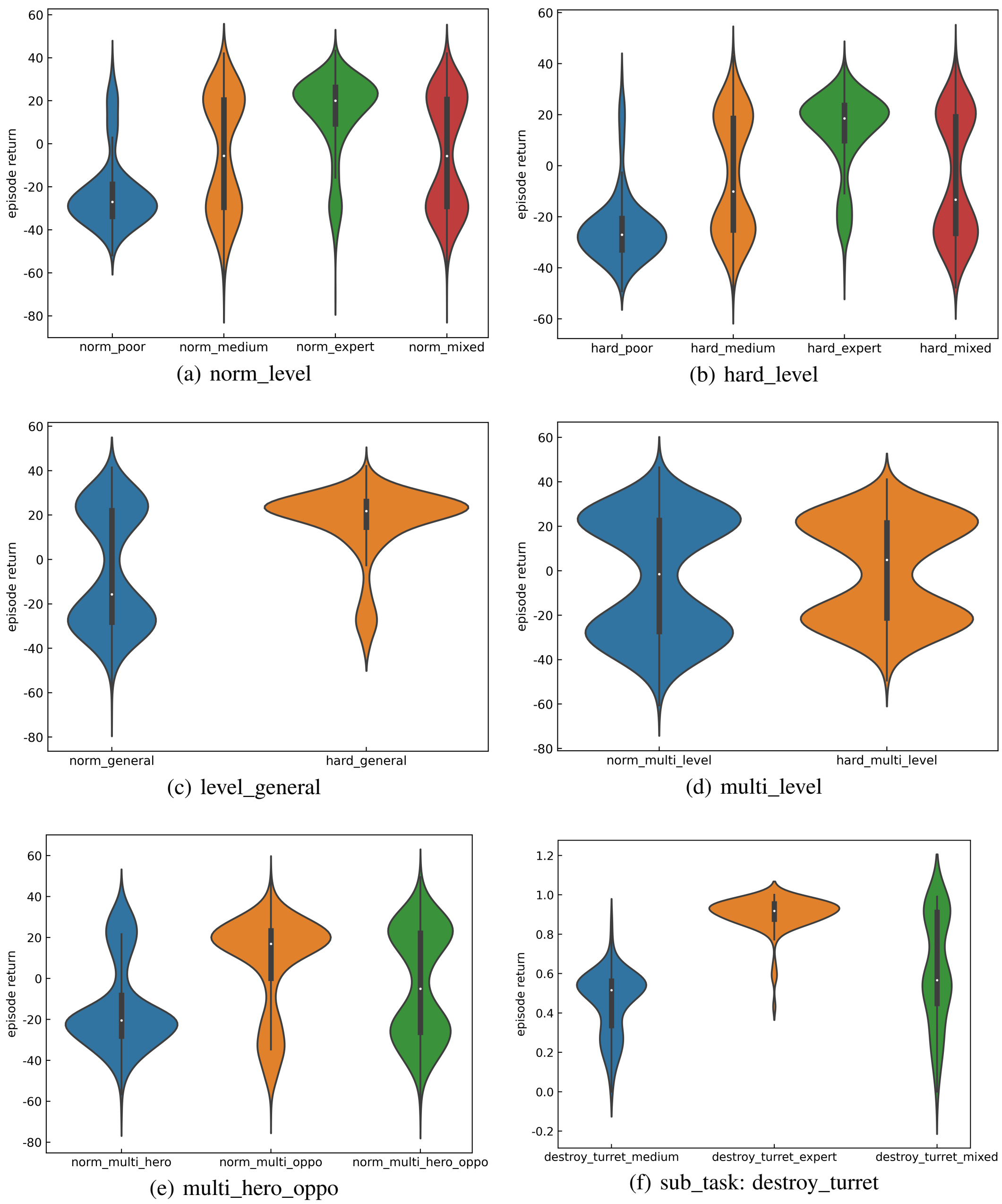
Table 4: Details of datasets in *Sub-Task: Destroy Turret*

Factors	Datasets/Experiments	Capacity	Heroes	Oppo_heroes	Average Score	Levels
Sub-Task	**destroy_turret_medium**	100	default	no	0.55	medium
	destroy_turret_expert	100	default	no	1.00	expert
	destroy_turret_mixed	100	default	no	0.73	-

$$normalized\_sub\_task\_score = \frac{random\_frame\_length - frame\_length}{random\_frame\_length - expert\_frame\_length} \tag{1}$$

C.2 HoK3v3

Table 5 presents the details of datasets in HoK3v3 game mode. The datasets are designed based on the design factors and collected by recording the battle trajectories of pre-trained models as described in Sec F. Typically, each dataset has a capacity of 1000 trajectories. The default heroes chosen for both camps are *{{zhaoyun}, {diaochan}, {liyuanfang}}*, with their Summoner Spells assigned as *{{smite}, {purify}, {purify}}* based on their respective roles. However, in specific scenarios such as



(a) norm_level (b) hard_level

(c) level_general (d) multi_level

(e) multi_hero_oppo (f) sub_task: destroy_turret

Figure 3: Violin diagrams of all datasets in HoK1v1.

Generalization or *Multi-Task* settings, we employ a random selection of heroes from a predefined set, *multi_hero*. This set comprises six heroes, with two heroes assigned to each role, namely *{{zhaoyun, zhongwuyan}, {diaochan, zhugeliang}, {liyuanfang, sunshangxiang}}*, with same Summoner Spells setting as in the *default* setup. Similarly, the win rate of the behavior policy is recorded in the column labeled *Win_rate* for reference. The column labeled *Levels* denotes the levels of opponents used for evaluation. Generally, a level of "1" is used for datasets with the "norm" prefix, while a level of "7" is used for datasets with the "hard" prefix. This distinction indicates varying levels of difficulty. The design of datasets and experiments pertaining to the concept of *Generalization* closely resembles that of the HoK1v1.

We introduce a sub-task called "*Gain Gold*" that builds upon the HoK3v3 game. In this modified version, we remove opponents and redefine the primary objective to focus on collecting gold within a limited number of time steps. This transforms the original competitive task into a resource collection task. Specifically, we set a maximum episode length of 8000 frames, and the controlled heroes are required to efficiently gather gold by killing monsters or creeps. As demonstrated in Table 6, we generate three datasets based on this sub-task, each consisting of 100 trajectories. The *gain_gold_medium* dataset is collected by a model with moderate performance, averaging 5904 gold collected. While, the *gain_gold_expert* dataset is obtained from a model with expert performance, averaging 12271 gold collected. Lastly, the *gain_gold_mixed* dataset combines the data from the previous two datasets equally. The scores are normalized based on Equation 2, where the $random\_gain\_gold$ is 5000 and $expert\_gain\_gold$ is 12000.

We have also generated violin charts in HoK3v3 to represent the distribution of episode returns in each dataset as shown in Fig. 4. The plot method used is similar to that in HoK1v1, with the exception of not using normalized scores in the *Sub-Task: Gain Gold*.

$$normalized\_sub\_task\_score = \frac{gain\_gold - random\_gain\_gold}{expert\_gain\_gold - random\_gain\_gold} \tag{2}$$

Table 5: Details of datasets in HoK3v3 game mode

Factors	Datasets/Experiments	Capacity	Heroes	Oppo_heroes	Win_rate	Levels
Multi-Difficulty	**norm_poor**	1000	default	default	16%	1
	norm_medium	1000	default	default	50%	1
	norm_expert	1000	default	default	82%	1
	norm_mixed	1000	default	default	49%	1
	hard_poor	1000	default	default	18%	7
	hard_medium	1000	default	default	50%	7
	hard_expert	1000	default	default	83%	7
	hard_mixed	1000	default	default	51%	7
Generalization	**hard_general**	1000	default	default	94%	8
	norm_general	1000	default	default	57%	5
	norm_hero_general	-	multi_hero	default	-	1
	hard_hero_general	-	multi_hero	default	-	7
	norm_oppo_general	-	default	multi_hero	-	1
	hard_oppo_general	-	default	multi_hero	-	7
Multi-Task	**norm_multi_level**	1000	default	default	50%	1
	hard_multi_level	1000	default	default	50%	7
	norm_multi_hero	1000	multi_hero	default	74%	1
	norm_multi_oppo	1000	default	multi_hero	26%	1
	norm_multi_hero_oppo	1000	multi_hero	multi_hero	50%	1
Heterogeneous	**norm_stupid_partner**	1000	default	default	50%	1
	norm_expert_partner	1000	default	default	50%	1
	norm_mixed_partner	1000	default	default	50%	1

Table 6: Details of datasets in *Sub-Task: Gain Gold*

Factors	Datasets	Capacity	Heroes	Oppo_heroes	Average Score	Levels
Sub-Task	**gain_gold_medium**	100	default	no	0.13	medium
	gain_gold_expert	100	default	no	1.04	expert
	gain_gold_mixed	100	default	no	0.58	-

Table 7: Details of sampling *generalization* datasets.

		Sampling and Training		Testing
Environments	**Datasets**	**Controlled side model**	**Opponent side model**	**Opponent side model**
HoK1v1	norm_general	level-1	level-0,2,4	level-1
	hard_general	level-5	level-0,2,4	level-5
HoK3v3	norm_general	level-5	level-1,4,7	level-5
	hard_general	level-8	level-1,4,7	level-8

D Environment Details

D.1 Honor of Kings Arena

For a more detailed account of the game settings, please refer to the original paper [38] and its documentation[6] of Honor of Kings Arena. In this context, we will only summarize the critical information that is relevant to the RL research.

• **Observation Space**

We have utilized the fundamental set of observations presented in the aforementioned paper [38]. Specifically, the observation space of Honor of Kings Arena consists of a normalized vector with 725 dimensions, which includes five main components: *hero_state_common_feature*, *hero_private_feature*, *creep_feature*, *turret_feature*, and *global_feature*. The details of the observation vector are demonstrated in Table 8. In the table, *Main_camp* and *Enemy_camp* refer to the information of the controlled side and enemy side, respectively. Moreover, the information of invisible units is set to the default value.

Table 8: Details of observation vector in Honor of Kings Arena

feature name	dimensions	description
Main_camp_hero_state_common_feature	102	hero's status, including whether it's alive, its ID and its health points (HP)
Main_camp_hero_private_feature	133	hero's specific kill information
Enemy_camp_hero_state_common_feature	102	enemy hero's status, including whether it's alive, its ID, its health points (HP)
Enemy_camp_hero_private_feature	133	enemy hero's specific kill information
Public_feature	14	visible information because of the turret
Main_camp_soldier_feature	18*4	the status of the creeps in a troop, including location and HP
Enemy_camp_soldier_feature	18*4	the status of the enemy's creeps in a troop, including location and HP
Main_camp_organ_feature	18*2	the status of turret and crystal
Enemy_camp_organ_feature	18*2	the status of enemy's turret and crystal
Global_feature	25	the period of the match

• **Action Space** To tackle the complicated control, the Honor of Kings adopt a structured action space. Specifically, illustrated in Fig. 5 the action space is 6 dimensions, consisting of a triplet form, i.e. the action button, the movement or skill offset and the target game unit, which covers all the possible actions of the hero hierarchically: 1) what action button to take, e.g. skill or move.; 2) who to target,

[6] `https://aiarena.tencent.com/hok/doc/`

e.g., a turret, an enemy hero, or a creep in the troop; 3) how to act, e.g., the discretized direction to move and release skills [38]. Please refer to Table 9 for details of action space in HoK1V1.

Table 9: Description of action space in HoK1v1

Action Class	Numbers	Description
Button	12	what action button to take, e.g. skill or move.
Move X	16	move direction along X-axis.
Move Y	16	move direction along Y-axis.
Skill X	16	skill offset along X-axis.
Skill X	16	skill offset along Y-axis.
Target	8	who to target, e.g., a turret or an enemy hero

• **Action Mask** There are two action masks designed to reduce the complexity of the action space, namely the *legal_action_mask* and the *sub_action_mask*. The former is constructed based on the rules of the game in order to exclude illegal actions, while the latter is determined by the selected button to eliminate actions that cannot be executed simultaneously with the chosen button, such as 'skill offset' and 'target unit' are not needed for 'move'.

• **Reward Design** The basic hero reward is a weighted average of several reward items, which is demonstrated in Equation 3. Subsequently, the hero's reward is transformed into a zero_sum value by subtracting the enemy's reward from it, as shown in Equation 4. Here, *team_reward* represents the average reward of the heroes within the team. Details of the reward items are demonstrated in Table 10.

.

$$\begin{aligned} hero\_reward = & w_1 * farming\_related + w_2 * KDA\_related + w_3 * damage\_related \\ & + w_4 * pushing\_related + w_5 * win/lose\_related \end{aligned} \tag{3}$$

$$hero\_reward_{zero\_sum} = hero\_reward - team\_reward_{enemy} \tag{4}$$

Table 10: Description of reward items in HoK1v1

Items	Type	Description
hp_point	dense	the rate of health point of hero
tower_hp_point	dense	the rate of health point of tower
money	dense	the increment of gold
ep_rate	dense	the rate of mana point
death	sparse	being killed
kill	sparse	killing an enemy hero
exp	dense	the increment of experience
last_hit	sparse	the lst hit for soldier

D.2 Honor of Kings 3v3 Arena

For a more detailed account of the game settings, please refer to the documentation of Honor of Kings 3v3 Arena (HoK3v3) [7]. The environment code of HoK3v3 is integrated into the open-source 1v1 code, both with official authorization from Honor of Kings [8].

• **Observation Space** Specifically, the observation space of HoK3v3 consists of a normalized vector with 4586 dimensions. The details of observation vector are presented in Table 11.

• **Action Space** The form of action space in HoK3v3 is similar to that in HoK1v1 while the number of actions is larger. Description of action space in HoK3v3 is presented in Table 12.

[7] `https://doc.aiarena.tencent.com/paper/hok3v3/latest/hok3v3_env/honor-of-kings/`
[8] `https://github.com/tencent-ailab/hok_env`

Table 11: Details of observation vector in HoK3v3.

feature name	dimensions	description
FeatureImgLikeMg	6*17*17	image-like feature, comprising six channels, which include barriers, grass, and other elements.
VecFeatureHero	6*251	the status of six heroes from the respective of controlled hero.
MainHeroFeature	44	private information of controlled hero.
VecSoldier	20*25	the status of all creeps.
VecOrgan	6*29	the status of turrets and crystals in both side.
VecMonster	20*28	the status of all monsters.
VecCampsWholeInfo	68	the status feature of the whole game.

Table 12: Description of action space in HoK3v3

Action Class	Numbers	Description
Button	13	what action button to take, e.g. skill or move.
Move	25	move direction.
Skill X	42	skill offset along X-axis.
Skill X	42	skill offset along Y-axis.
Target	39	who to target, e.g., a turret or an enemy hero

- **Reward Design** The basic hero reward is a weighted average of several reward items. Then the reward of each hero is processed to be zero_sum in minus the team reward of enemy which is the average of the hero rewards of 3 enemy heroes. The details of reward items are demonstrated in Table 13

Table 13: Description of reward items in HoK3v3

Items	Type	Description
hp_rate_sqrt_sqrt	dense	the fourth root of the rate of health point of hero
money	dense	the increment of gold
exp	dense	the increment of experience
tower	dense	the rate of health point of turrets
killCnt	sparse	kill an enemy
assistCnt	sparse	assisting in the termination of an adversary
deadCnt	sparse	being killed
total_hurt_to_hero	dense	damage dealt to the enemies
atk_monster	dense	attack an monster
atk_crystal	dense	attack the crystal of enemy
win_crystal	sparse	destroy the crystal of enemy

E Framework APIs

We provide an example of the APIs in our framework, Listing 1. A comprehensive account of our framework can be found in our readily accessible open-access code repository.

F Evalution Protocols: Multi-Level Models

Based on the parallel training system named **SAIL** proposed by previous work [45], we have extracted and published several checkpoints from pre-trained dual-clip PPO [45, 44] models with varying levels determined by the outcome of the battle separately for HoK1v1 and HoK3v3.

Here, we present tables displaying the win rate of each level model against the model listed directly below it. The win rate is calculated with fixed hero selection, i.e. *luban* for HoK1v1 and *{zhaoyun,diaochan,liyuanfang}* for HoK3v3, which may not right for other hero selection. Table 14 presents the win rate in the HoK1v1, where the *win_rate* column represents the win rate of *model1* against *model2*. Table 15 displays the win rate in HoK3v3. Additionally, we have included an API in our framework that allows researchers to conveniently test the win rate between any levels.

Table 14: Win rate of multi-level models in HoK1v1

model1	model2	win_rate
1v1_level_1	1v1_level_0	88%
1v1_level_2	1v1_level_1	79%
1v1_level_3	1v1_level_2	59%
1v1_level_4	1v1_level_3	97%
1v1_level_5	1v1_level_4	70%
1v1_level_6	1v1_level_5	73%
1v1_level_7	1v1_level_6	70%

Table 15: Win rate of multi-level models in HoK3v3

model1	model2	win_rate
3v3_level_1	3v3_level_0	97%
3v3_level_2	3v3_level_1	83%
3v3_level_3	3v3_level_2	50%
3v3_level_4	3v3_level_3	65%
3v3_level_5	3v3_level_4	63%
3v3_level_6	3v3_level_5	59%
3v3_level_7	3v3_level_6	80%
3v3_level_8	3v3_level_7	82%

G Additional Experimental Details

G.1 Additional Algorithm Details

- **Encoder**: Due to the complexity of the observation space, it is necessary to utilize a well-designed encoder for effective feature extraction. Taking inspiration from the "divide and conquer" approach employed in previous works [45, 44], in each algorithm, we implement a shared encoder network to process features, instead of directly feeding raw observations into the policy or critic network. For further details on the design of the encoder network, please refer to the mentioned papers [45, 44] as well as our code.

- **BC**: Behavior clone with maximum likelihood estimation loss. While, in multi-agent setting, HoK3v3, we adopt shared parameter and independent learning paradigm [34].

- **CQL** [19]: The implementation of Conservative Q-Learning is based on the original version [19] designed for discrete action spaces[9].

- **QMIX+CQL**: Due to the decoupling of control dependencies [45], the action space of HoK is structured with multi-head, which is similar to the joint action space in multi-agent settings. Inspired

[9] `https://github.com/aviralkumar2907/CQL/tree/master/atari`

by this, we propose QMIX-CQL by incorporating mixer in QMIX [29] with CQL and use global Q to calculate td error term and use local Q to calculate CQL-loss term.

- **TD3+BC** [7]: Our implementation of TD3-BC is based on the open-source code[10]. In addition, the policy network and critic network share an encoder, which is updated simultaneously by both losses. Besides, We utilize Gumbel-Softmax reparameterization method to generate discrete actions for TD3 [8].

- **IQL** [16]: We implement IQL based on the open-source pytorch version[11]. The network design is similar to TD3-BC except for an additional value network.

- **IND+CQL** and **COMM+CQL**: To accommodate a multi-agent setting, based on the implementation of CQL, we adopt the independent learning paradigm and shared parameters, referred to as IND-CQL. Additionally, we introduce COMM-CQL which adds communication between agents by means of shared information that is constructed using max pooling.

- **IND+ICQ** and **MAICQ** [43]: We implement IND+ICQ and MAICQ based on the original published code[12]. IND+ICQ adopts independent learning paradigm, while MAICQ adopts CTDE paradigm by decomposing the joint-policy under the implicit constraint. The actor and critic networks update the shared encoder simultaneously as TD3-BC.

- **OMAR** [28]: The open-access code[13] of OMAR is not suitable for a discrete action space. Consequently, based on the core idea of it, we have undertaken the task of re-implementing OMAR to accommodate a discrete version.

G.2 Hyperparameters

We have compiled the hyperparameters of HoK1v1 and HoK3v3 in Tables 16 and 17, respectively. These tables encompass the parameters of the training process, algorithm and optimizer settings.

Regarding the computing resources employed in HoK1v1, we utilize the Tesla T4 GPU and the AMD EPYC 7K62 48-Core Processor CPU. For the sampling process, 50 CPU cores are utilized, and each dataset required approximately 30 to 40 minutes for sampling. During the training process, each training experiment is conducted with one Tesla T4 GPU and two CPU cores, with an average training time of 9 hours per seed for 500000 training steps.

Regarding the computing resources employed in HoK3v3, we utilize the Tesla T4 GPU and the AMD EPYC 7K62 48-Core Processor CPU. For the sampling process, 50 CPU cores are utilized, and each dataset required approximately 80-90 minutes for sampling. During the training process, each training experiment is conducted with one Tesla T4 GPU and four CPU cores, with an average training time of 20 hours per seed for 500000 training steps.

Consequently, a total of 14 GPUs and 552 CPU cores are used to accommodate the overall computation requirements.

G.3 Additional Results Discussion

- **Why is the performance of baseline models in the HoK1v1 comparatively inferior to those in the HoK3v3 setting?**

The experimental results conducted in the HoK1v1 reveal that the performance of baseline models is comparatively inferior to those in the HoK3v3 setting. This disparity can be attributed to the higher level of adversarial conditions present in the HoK1v1 environment. Furthermore, within the context of HoK3v3, if one teammate makes a sacrifice during a battle, the remaining two teammates are able

[10] `https://github.com/sfujim/TD3_BC`
[11] `https://github.com/gwthomas/IQL-PyTorch`
[12] `https://github.com/YiqinYang/ICQ/tree/5a4da859ef597005040f79128ee6163547cf178d`
[13] `https://github.com/ling-pan/OMAR`

Table 16: Hyperparameters for HoK1v1. The values of hyperparameters for algorithms are derived from their original implementation.

Hyperparameters	**Value**
Batch Size	128
γ	0.99
Max Steps (Exclude *Sub-Task* Datasets)	500000
Max Steps (*Sub-Task* Datasets)	100000
LSTM Time Steps	16
τ (Soft-Target-Update)	0.005
num_threads	2
final_evaluation_episodes	150
CQL α	10.0
TD3+BC α	2.5
IQL τ	0.7
IQL β	3.0
Optimizer	Adam
beta1	0.9
beta2	0.999
eps	1.00E-08
Learning Rate	3.00E-04

Table 17: Hyperparameters for HoK3v3. The values of hyperparameters for algorithms are derived from their original implementation.

Hyperparameters	**Value**
Batch Size	512
γ	0.99
Hard Update Frequency	2000
Max Steps	500000
Max Steps (*Sub-Task*)	100000
Iteration Steps	1000
Buffer Workers	2
num_threads	4
final_evaluation_episodes	150
CQL α	10.0
ICQ critic β	1000
ICQ policy β	0.1
OMAR coe	0.5
Optimizer	Adam
beta1	0.9
beta2	0.999
eps	1.00E-08
Learning Rate	1.00E-04

to maintain their collaboration and continue to fight. This aspect ensures a greater level of robustness in the HoK3v3 when compared to the HoK1v1.

- **What are the reasons behind the underperformance of TD3+BC and OMAR?**

TD3+BC and OMAR demonstrated subpar performance in HoK1v1 and HoK3v3, respectively. The main cause of their lackluster outcomes stems from the fact that TD3 [8], upon which TD3+BC and OMAR are built, is incompatible with discrete action spaces. To enhance OMAR's performance, we replaced TD3 with advantage-weighted BC. This modification resulted in performance improvements.

- **QMIX+CQL in HoK3v3.**

We also implemented **QMIX+CQL** in the HoK3v3 game mode by adopting an independent learning paradigm. We thoroughly validated the performance of **QMIX+CQL** and compared it with **IND+BC** and **IND+CQL**, aggregating the results in Table 18. It is demonstrated that **QMIX+CQL** exhibits superior performance compared to **IND+CQL**, indicating that our novel method is also suitable for multi-agent settings with a structured action space.

Table 18: Validation of **QMIX+CQL** in HoK3v3 game mode.

Factors	Datasets	IND+BC	IND+CQL	QMIX+CQL
Multi-Level	norm_poor	0.1±0.01	0.03±0.01	**0.11±0.03**
	norm_medium	0.48±0.01	0.4±0.03	**0.52±0.04**
	norm_expert	0.52±0.03	0.84±0.06	**0.85±0.04**
	norm_mixed	0.35±0.25	0.46±0.12	**0.47±0.29**
	hard_poor	**0.16±0.03**	0.12±0.03	0.13±0.02
	hard_medium	**0.38±0.05**	0.31±0.02	0.37±0.08
	hard_expert	0.65±0.01	0.67±0.04	**0.7±0.03**
	hard_mixed	0.32±0.14	0.3±0.1	**0.43±0.04**
Generalization	norm_general	**0.34±0.05**	0.29±0.09	**0.34±0.02**
	hard_general	0.28±0.03	0.28±0.04	**0.32±0.01**
	norm_multi_hero_general	0.17±0.03	0.14±0.04	**0.18±0.06**
	hard_multi_hero_general	0.16±0.05	0.17±0.02	**0.19±0.02**
	norm_multi_oppo_general	**0.21±0.01**	0.14±0.04	0.14±0.03
	hard_multi_oppo_general	0.09±0.06	0.09±0.02	**0.1±0.02**
Multi-Task	norm_multi_level	0.43±0.09	0.34±0.04	**0.45±0.02**
	hard_multi_level	**0.38±0.08**	0.29±0.07	0.35±0.04
	norm_multi_hero	**0.57±0.07**	0.3±0.07	0.56±0.13
	norm_multi_oppo	**0.09±0.04**	0.07±0.03	**0.09±0.02**
	norm_multi_hero_oppo	**0.3±0.04**	0.26±0.07	**0.3±0.03**
Heterogeneous	norm_stupid_partner	0.11±0.15	0.24±0.17	**0.55±0.05**
	norm_expert_partner	0.36±0.09	0.57±0.04	**0.72±0.07**
	norm_mixed_partner	0.49±0.2	0.32±0.03	**0.66±0.05**
Sub_Task	gain_gold_medium	**0.13±0.01**	0.12±0.01	**0.13±0.02**
	gain_gold_expert	1.01±0.03	1±0.01	**1.02±0.01**
	gain_gold_mixed	**0.64±0.29**	0.41±0.1	0.58±0.17

H Additional Discussion

H.1 The significance of our design factors in the context of offline reinforcement learning

Task difficulty: Intuitively, the level of difficulty in the environment significantly impacts the performance of algorithms. However, previous researches only utilized one set of datasets with a uniform level of difficulty in the environment. Providing datasets with diverse difficulty can not only more comprehensively evaluate the ability of offline algorithms but also be more suited for real-world tasks like HoK, which are characterized by diverse levels of difficulty.

Multi-task: Combining offline reinforcement learning with multi-task learning enables efficient use of limited data. Sharing knowledge[1] and representations across tasks enhances data efficiency, leading to more general and robust feature learning. Besides, multi-task learning facilitates knowledge transfer between tasks. Leveraging shared parameters and representations accelerates learning for the target task in offline reinforcement learning, benefiting from related tasks' knowledge.

Generalization: Firstly, in offline RL, learning is based on a fixed dataset collected from previous experiences. This dataset might not cover all possible scenarios, so the learned policy needs to generalize well to new, unseen situations to perform effectively. Secondly, real-world environments are often complex and diverse. A policy only limited to the dataset without generalizing would likely fail when facing even slightly different conditions. Generalization ensures the policy's adaptability to various situations in the real-world scenarios.

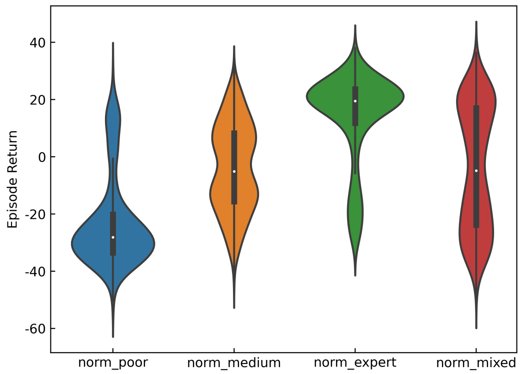
Heterogeneous Teammate: Heterogeneous teammates are a crucial research direction in Multi-Agent Reinforcement Learning (MARL). In practical scenarios such as HoK or other multi-agent systems, players typically possess varying capacities. Consequently, the datasets collected from real-world scenarios consist of heterogeneous teammate data, necessitating the need for corresponding research in the offline MARL domain.

H.2 From the perspective of the Honor of Kings game, why offline reinforcement learning is necessary and what potential limitations exist when compared to online reinforcement learning?

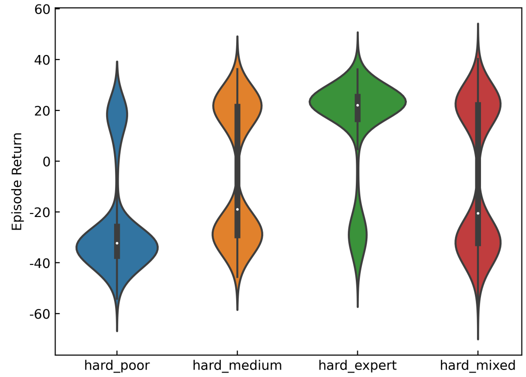
Training agents for the Honor of Kings game using offline reinforcement learning (RL) offers several advantages, including reduced training time, lower computation resource requirements, and better utilization of existing data resources. We compare the computational and time costs of online RL and offline RL in Table 19. It is evident that training an online agent from scratch to reach specific levels (level_5 for HoK1v1 and level_7 for HoK3v3) requires thousands of CPU cores and dozens of hours. On the other hand, training offline agents to reach same levels only requires a few CPUs and a shorter training time using pre-collected datasets. Additionally, there is a wealth of previously collected battle data that can be used for training offline RL agents. However, compared to online RL, it is important to note that offline RL in the HoK game heavily relies on large amounts of high-level battle data to train expert-level agents, which may be a potential limitation.

Table 19: The comparison of the computational and time costs between online RL and offline RL.

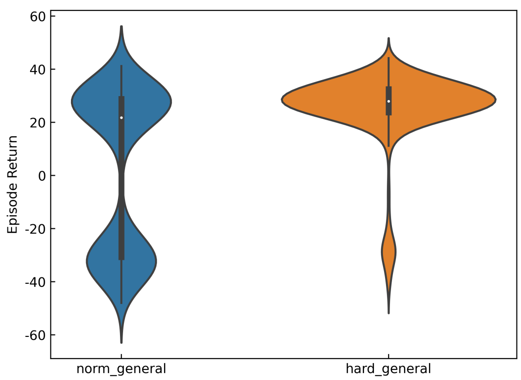
Online/Offline	**CPU cores**	**GPU cores**	**Performance**	**Training time (hours)**
HoK1v1 (online)	4000	2	1v1_level_5	60h
HoK1v1 (offline)	2	1	1v1_level_5	9h
HoK3v3 (online)	1000	1	3v3_level_7	97h
HoK3v3 (offline)	4	1	3v3_level_7	20h



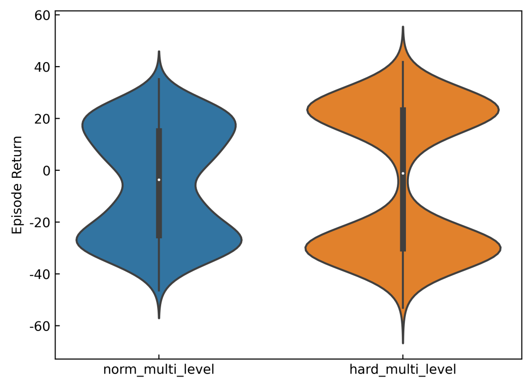
(a) norm_level



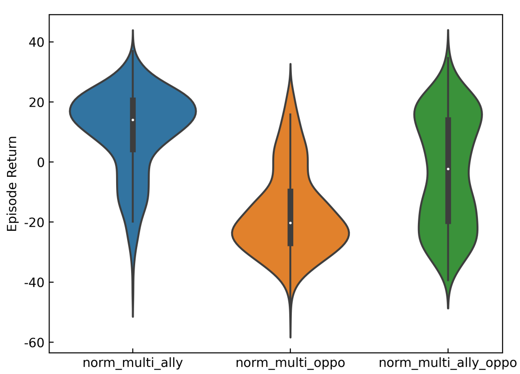
(b) hard_level



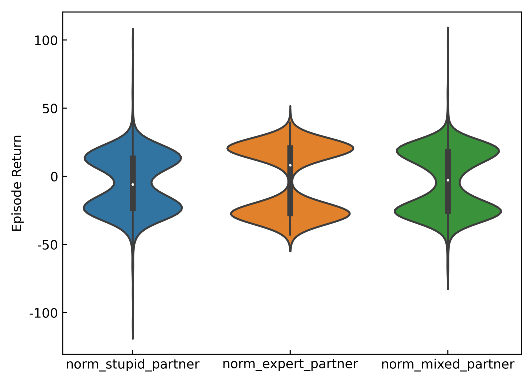
(c) level_general



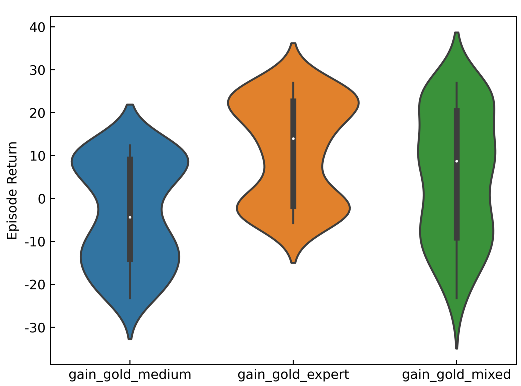
(d) multi_level



(e) multi_hero_oppo



(f) heterogeneous



(g) sub_task: gain_gold

Figure 4: Violin diagrams of all datasets in HoK3v3.

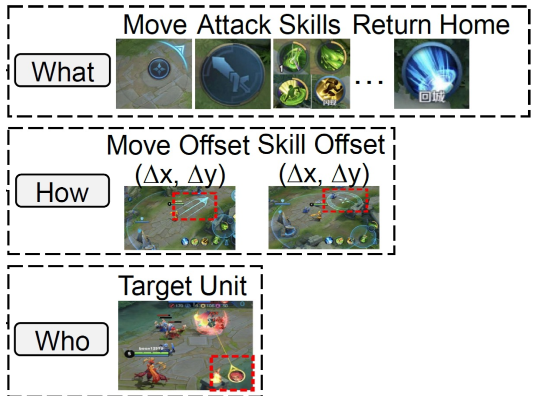


Figure 5: Action space in HoK1v1 [38]

```
cd  <root_path>

# sample example
sh offline_sample/scripts/start_sample.sh <args>

# train example
python offline_train/train.py --<args>

# evaluate example
python offline_eval/evaluation.py --<args>

```

Listing 1: APIs example