entropy

MDPI

Article

KnowRU: Knowledge Reuse via Knowledge Distillation in Multi-Agent Reinforcement Learning

Zijian Gao, Kele Xu *, Bo Ding and Huaimin Wang

College of Computer, National University of Defense Technology, Changsha 410000, China; gaozijian19@nudt.edu.cn (Z.G.); dingbo@aliyun.com (B.D.); whm_w@163.com (H.W.)
* Correspondence: kelele.xu@gmail.com; Tel.: +86-166-7316-1118

Abstract: Recently, deep reinforcement learning (RL) algorithms have achieved significant progress in the multi-agent domain. However, training for increasingly complex tasks would be time-consuming and resource intensive. To alleviate this problem, efficient leveraging of historical experience is essential, which is under-explored in previous studies because most existing methods fail to achieve this goal in a continuously dynamic system owing to their complicated design. In this paper, we propose a method for knowledge reuse called "KnowRU", which can be easily deployed in the majority of multi-agent reinforcement learning (MARL) algorithms without requiring complicated hand-coded design. We employ the knowledge distillation paradigm to transfer knowledge among agents to shorten the training phase for new tasks while improving the asymptotic performance of agents. To empirically demonstrate the robustness and effectiveness of KnowRU, we perform extensive experiments on state-of-the-art MARL algorithms in collaborative and competitive scenarios. The results show that KnowRU outperforms recently reported methods and not only successfully accelerates the training phase, but also improves the training performance, emphasizing the importance of the proposed knowledge reuse for MARL.

Keywords: multi-agent reinforcement learning; knowledge reuse; knowledge distillation

Citation: Zijian, G.; Xu, K.; Ding, B.; Huaimin, W.; KnowRU: Knowledge Reuse via Knowledge Distillation in Multi-Agent Reinforcement Learning. *Entropy* **2021**, *23*, 1043. https://doi.org/10.3390/e23081043

Academic Editor: Sotiris Kotsiantis

Received: 6 July 2021
Accepted: 8 August 2021
Published: 13 August 2021

1. Introduction

Reinforcement learning (RL) has made great progress in solving complicated tasks, such as Atari games [1], board games [2], and video-game playing [1]. With the compelling performance of single-agent models, multi-agent RL (MARL) tasks, such as collaboration and competition among multiple agents, have piqued the interest of researchers in several fields [3,4], as the applications of MARL seems to be evident.

Current MARL algorithms are still highly task-specific and lack the ability to generalize to new environments. Moreover, for resource-limited embedded systems, training the MARL system from scratch would be extremely time-consuming and resource intensive because of the MARL system's high complexity. Efficient transfer and use of knowledge between tasks can alleviate the aforementioned issues; sustainable efforts have been made in this field. One category of solutions employs the transfer learning paradigm to reuse the knowledge of historical experience, which can relieve the burden of training a new model with previous experience [5].

However, most existing transfer learning methods for multi-agents mainly depend on the hand-coded design, which requires knowledge from domain experts. For example, a method based on the Pepper algorithm [6] proposed by [7] is used to obtain strategies for opponents in adversarial scenarios and calculate policies against them. Then, agents in new scenarios evaluate the opponents' strategy and reuse the knowledge by learning from the calculated policy. Similarly, a genetically programmed approach [8] utilizes strategies from previously trained networks to new tasks. A set of neural networks are trained to predict the value of each action and obtain a set of action strategies to build a multi-tiered

architecture for agents to learn when to trigger which strategy in new tasks. However, these approaches mainly rely on hand-coded design, which is closely related to specific tasks, such as the mapping of state variables or modeling of opponents. Obviously, owing to the difficulties in professional and hand-coded design for tasks, existing methods that require sufficient domain knowledge are not universal enough and are difficult to deploy. It is desirable to reuse knowledge from previous experience using a method that can be widely used and easily deployed without considering how and what to transfer.

In this paper, we propose a method for knowledge reuse called KnowRU, which can be easily deployed in MARL algorithms. KnowRU can shorten the training phase for new tasks while improving the asymptotic performance of agents. Our motivation is derived from knowledge distillation (KD) [9], which has been successfully applied in the field of computer vision. By leveraging the knowledge of well-trained agents in previous tasks as a historical experience, we employed the KD approach to transfer knowledge to agents for new tasks. Here, we suppose that the action taken by the agent in response to the environment is the simplest form of knowledge, and that action is determined by the network's output. Therefore, mimicking the output is a potentially feasible way to reuse the knowledge of historical agents. During the training phase of new tasks, agents not only achieve higher rewards in the environment, but also mimic the outputs of previous agents. In this way, agents can learn from varying rewards and derive knowledge by mimicking historical agents. Knowledge from historical agents can be viewed as a fundamental consensus among different tasks owing to discrepancies in the tasks. Empirical experiments were conducted on different tasks and MARL algorithms to validate the effectiveness of KnowRU. Figure 1 illustrates an example in which agents must work together to reach the closest target as soon as possible. We retained the well-trained agents from the previous task; agents in the target task can observe how the well-trained agents would work in the same situation and transfer knowledge by mimicking the observed actions.

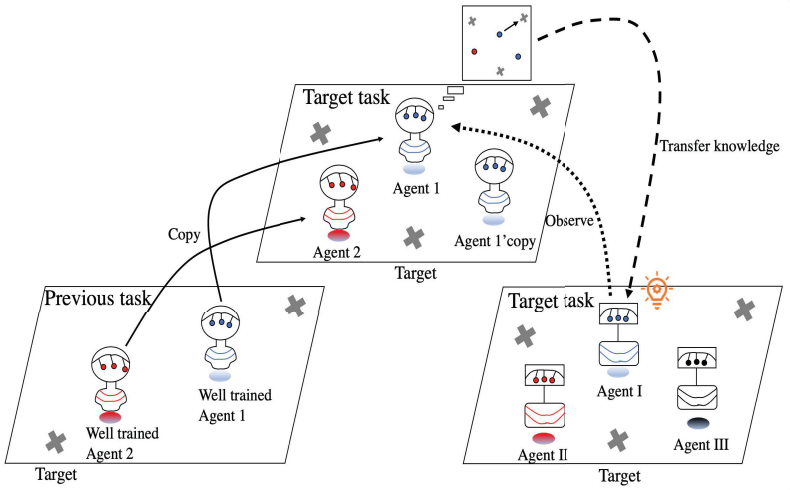


Figure 1. Overview of KnowRU. Agents in the target task mimic historical agents' actions in the same situation, and then effectively reuse the knowledge in the previous task and allow further learning in a new environment.

The contributions in this paper are as follows:

- We propose a task-independent KD framework for MARL, which focuses on not only accelerating the training phase, but also improving asymptotic performance for new tasks.
- Different strategies are explored to further improve the knowledge transfer performance.
- Extensive empirical experiments demonstrate the effectiveness of KnowRU in different task scenarios with different MARL algorithms.

2. Related Work

A multi-agent system (MAS) can be defined as stochastic games (Markov games, SG) [10], which extend from basic Markov decision processes (MDPs). Based on these theories, many excellent MARL algorithms have been proposed in recent years. The conventional approaches developed for basic MDPs, such as Q-Learning [11] and policy gradient [12], fail to train agents well in MAS because they have no head for considering environmental dynamics due to non-stationary policies of other agents. Instead, the MARL algorithms based on the actor–critic architecture [13], such as multi-agent deep deterministic policy gradient (MADDPG) [14], consider all agents' information with a centralized critic. Furthermore, MAAC [15] learns a centralized critic with an attention mechanism to help agents focus on vital information.

However, because of the high sample complexity of conventional RL methods, training agents from scratch every time is difficult, especially in a continuous variational task, which would cost a considerable amount of time and computational resources. Transfer learning [16] is a practical approach to alleviate the problem via knowledge reuse. The vanilla MARL algorithms [17–20] focus only on designing some mechanism to improve the learning process of the current task. There are some methods aimed at mapping the relationships among tasks. For example, ref. [21] used the task relations to aggregate specific strategies in source tasks and generate an abstract policy that is used to help agents quickly adapt to new tasks. However, the indescribable relations among tasks are the difficulty of the abstracting policy. The use of evolutionary algorithms to transfer knowledge in an evolved multi-agent system is a feasible approach. Ref. [22] presented a neuro-evolutionary method that uses a neural network to codify agents' policies, optimizes the network's topology, and weights though interactions with respect to the environment. However, the method relies heavily on humans to define parameters and mapping between tasks. These above-mentioned transfer learning approaches differ from our method in the following ways. (1) They mainly focus on how and what to transfer between tasks. (2) The nifty artificial design for specific tasks based on sufficient domain knowledge is the key to their success. In this study, KnowRU only requires a policy model related to target tasks, without considering the model structure, relationships between tasks, or task-specific design. In comparison, KnowRU shows wider application prospects and can be aggregated into more MARL algorithms in a more convenient way.

KD is a type of knowledge transfer method, which was first proposed by [23] and has since become popular [9]. KD compresses the knowledge of large-scale complex models (*teachers*) into small and efficient models (*students*) to facilitate the deployment of models on devices with insufficient computing resources. The idea of KD inspired us to train the small student model with not only true labels but also soft targets provided by the well-performed large teacher model. Currently, the main KD methods can be divided into three categories: logits-based methods for learning the output layer, feature-based methods for learning the hidden layers, and relation-based methods for learning the relations between network layers. Some studies were also conducted on the application of KD in RL [24,25]. They mainly focus on making use of agent-level knowledge to tackle the problems in a single RL task with a KD paradigm. However, our primary focus was knowledge reusability in multi-agent tasks and to solve the problem of agents' rapid adaptation to the dynamic changes of tasks in experiments.

3. Methodology

3.1. Preliminaries and Notations

MDPs [5] in MARL can be denoted as a tuple, $< S, U, T, R_{1\ldots n}, \gamma >$, where S is the state space, U is the joint action space, T is the state transition function, R_i is the reward function of agent i, γ is the discount factor, and n is the number of agents. The observation of agent i is in the current state S_i is O_i; then, the agent takes action A_i with policy π_{θ_i}, and the environment produces the next state according to T. Agent i can obtain rewards R_i

from the environment according to state S_i and action A_i. Agents simply aim to take proper actions that lead to maximal total return, $R = \sum_{t=0}^{T} \gamma^t R_i^t$, where T is the time horizon.

Actor–Critic (AC): To overcome high variance of policy gradient, the actor–critic [13] methods use a function as the critic to evaluate actions and replace the return term of the policy gradient with the value function. In RL, given the state and action, according to the Bellman equation, the critic function can be written as Equation (2), and the returns can be estimated so that the actor can be updated at every step. In this way, the models can be updated with less noise. The AC framework lays down a solid foundation for future multi-agent algorithms. Let π in Equation (2) be the agent's policy.

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]]. \tag{1}$$

MADDPG: MADDPG [14] is an extension of the deep deterministic policy gradient (DDPG) [26], which uses "target+online" networks and the experience replay to deal with the failure of AC in continuous action space. The critic in vanilla DDPG focuses only on the local information of the current agent, rather than the global view, resulting in a not-so-stable performance in multi-agent systems. Instead, MADDPG exploits global observations and actions for all agents with a centralized action-value function:

$$Q_i^{\pi}(\mathbf{x}, a_1, \ldots, a_N), \tag{2}$$

where $\mathbf{x} = (o_1, \ldots, o_N)$ and i denotes the current agent. Let $\pi = \{\pi_1, \ldots, \pi_N\}$ be the set of all agent policies. If all actions taken by all agents are accessible, the learning processes conform to the Markov property. This is why MADDPG works well in multi-agent systems.

MAAC: MAAC [15] makes a significant contribution to the AC-based MARL algorithm with the attention mechanism. Every agent queries information from other agents' observations and actions, and then estimates its value based on the information. The Q-value function, Q_i^{ψ}, considers the current agent i's observation, action, and other agents' contributions as follows:

$$Q_i^{\psi}(o, a) = f_i(g_i(o_i, a_i), x_i), \tag{3}$$

where f_i is a multilayer perceptron (MLP), and g_i is an MLP embedding function. The contribution from the key-value memory model is a weighted value of other agents:

$$x_i = \sum_{j \neq i} \alpha_j v_j = \sum_{j \neq i} \alpha_j h\big(V g_j(o_j, a_j)\big), \tag{4}$$

where v_j denotes a function of agent j's embedding that is encoded with an embedding function, and then linearly transformed by shared matrix V; h is an element-wise nonlinearity; and α_j represents the attention weight.

In this study, we are in line with the AC methods and consider MADDPG and MAAC as our baselines.

3.2. KD

As mentioned earlier, because the teacher model provides more useful information for the student model, KD has achieved success in the field of computer vision. The soft probability output of trained teachers is key to distillation. Let l_t be the input log of the final softmax layer of the teacher network, where $l_t = [l_1, l_2, ..., l_j]$. The logits are converted into probabilities $q_t = [q_1, q_2, ..., q_j]$ using the following softmax function: $q_i = \frac{e^{l_i}}{\Sigma_j e^{l_j}}$ where i represents the i-th neuron. To extract more information compared with true labels, [9] proposed softening the teacher probabilities with temperature T:

$$q_i = \frac{exp(l_i/T)}{\Sigma_j exp(l_j/T)}. \tag{5}$$

In KD, such dark knowledge from the soft output of the teacher provides more information than that from the true labels. Based on the same image input x, the teacher and student networks produce probabilities $q_t(x)$ and $q_s(x)$ with Equation (5), respectively. The gap between $q_t(x)$ and $q_s(x)$ is usually penalized by the Kullback–Leibler (KL) divergence (Equation (7)) or cross-entropy loss:

$$\mathcal{L}_{KD} = T^2 KL(q_s(x), q_t(x)). \tag{6}$$

$$KL(P\|Q) = \sum P(x) \log \frac{P(x)}{Q(x)} \tag{7}$$

where *P(x)* and *Q(x)* are two probability distributions of the random variable x. Temperature T in Equation (6) also aims to soften the output of the teacher network. Then, through back propagation of L_{KD}, the student network could reuse knowledge from the teacher network. KD inspires us to believe that minimizing the gap between previous agents and current agents through the skill of distillation is the essence of knowledge reuse. We then draw upon the KD thought in our MARL research to reuse knowledge and verify the feasibility of KnowRU in Section 3.3.

3.3. Knowledge Reuse via Mimicking

Consider a real-world scenario in which our training task is constantly variational and the number of agents is also increasing. It is impractical to train agents from scratch whenever a task changes due to the high cost of time and resources. There are some similarities between tasks, and the knowledge learned in previous tasks can be regarded as historical experiences. Thus, knowledge reuse from historical experiences is important. However, we have argued that some existing works cannot reuse knowledge directly and effectively without exquisite design or expert-level experience. To solve such problems as rapid adaptation to dynamic changes in the number of agents, it is necessary to find an easier and more practical way of reusing knowledge. We aimed at designing a method that works well in such scenarios.

In this study, based on the concept of KD, we propose to reuse knowledge through mimicking to minimize the gap between tasks in MAS. We designed a task-independent knowledge reuse method that can be applied based on multiple MARL algorithms without a task-specific design. First, we used the policy models of the well-trained agents homogeneous with the current training agents in previous tasks related to the current task. Then, we paired every current agent and every homogeneous policy model. If the number of current agents is greater than the number of policy models, repeated pairings are allowed. During the training process, the same observations of the current agent were input into both the previous policy and current models. Based on the same input, the logits, which are the inputs of the last softmax layer, are used to measure the gap between tasks with a loss function. Simultaneously, current agents receive feedback from the environment through returns. It is worth mentioning that using logits and softmax is more of a direct and simple approach but using other characteristics of neural networks to measure the gap is also feasible.

In this way, the agents of new tasks are trained by not only maximizing the total return from the environment but also by mimicking the output of previously well-trained policy models. Finally, considering the differences between tasks, determining how to reasonably combine the mimicking gap loss and returns from the environment is essential for training. Here, we used hyperparameter α to adjust the relationship between these two factors and finally obtain the total loss, $\mathcal{L}_{all}$, for back propagation. Figure 2 illustrates the main components of KnowRU based on the AC framework that is widely used in MARL. KnowRU has been proven to be feasible in various experimental scenarios in Section 4.

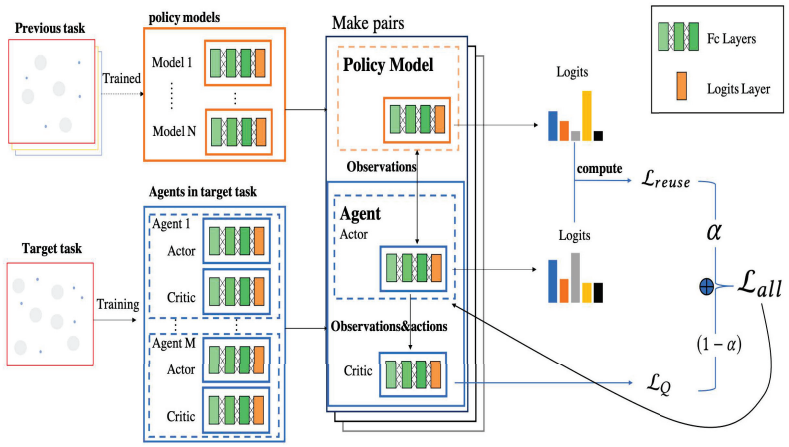


Figure 2. Workflow of KnowRU based on the actor–critic algorithm. It can be explained as three steps: 1: Match the policy models in the previous task with the agents in the target task. 2: Calculate the loss from the environment and the gap between the actor network and policy model. 3: While the agent receives feedback from the environment, it also mimics the actions of the policy model.

Mimicking. In the field of computer vision, the teacher model used to transfer knowledge is a well-trained model that is trained on the same dataset as the current student model. Therefore, to extract dark knowledge, it is necessary to soften the teacher model's output probability using Equation (5) and minimize the gap between them with cross-entropy (CE) loss or KL loss, using Equation (6). However, in the MARL, the source and target tasks might be different. The previous policy models may be overconfident or not authoritative in the new task. Therefore, it is not necessary to soften the probability of the policy model with hyperparameter T. Assume that a_p and a_c are the input logits of the final softmax layer in the previous policy model and current model, where $a_p = [a_1, a_2, ..., a_n]$ and $a_c = [a'_1, a'_2, ..., a'_n]$, respectively. Here, we use the mean-squared error (MSE) loss as the $\mathcal{L}_{reuse}$ loss function.

$$\mathcal{L}_{MSE} = \frac{1}{n}\sum_{i=1}^{n}(a_i - a'_i)^2. \tag{8}$$

Task's guide and specialization. We state that the previous policy model and current model usually work for different but similar tasks in MARL. The previous policy model cannot completely bootstrap the current learning; however, there is still some knowledge between similar tasks that can be described as a consensus. Some studies [27,28] showed that unprincipled reuse of knowledge may not help but hinder training. Therefore, it is vital to reuse the consensus in a reasonable manner. The training process is divided into two phases: *phaseI* **guide** and *phaseII* **specialization**. In *phaseI*, the previous policy model guides the current model to reuse knowledge. As the training progresses, the agents enter into the stage of **specialization,** stepwise, and the difference between tasks gradually increases. As shown in Figure 2, the training depends on two factors: $\mathcal{L}_{reuse}$ and $\mathcal{L}_Q$. We used hyperparameter α to adjust the weights of the two factors and achieve the transition between the two stages.

$$\mathcal{L}_{all} = \alpha\mathcal{L}_{reuse} + (1-\alpha)\mathcal{L}_Q, \tag{9}$$

where $\alpha \in [0, 1]$. Here, $\mathcal{L}_{reuse} := \mathcal{L}_{MSE}$, where $\mathcal{L}_Q$ is defined as the Q-value obtained from the action-value function in the algorithms. By controlling α, we simulated the shift of focus from the guide phase to the specialisation phase in the learning process. In our experimental settings, α usually starts at around 0.5 and decreases linearly to 0.02. Some studies [29] showed that the low weight of knowledge reuse left behind can provide some noise for training to avoid task overfitting. It is worth mentioning that $\mathcal{L}_{reuse}$ and $\mathcal{L}_Q$ may not be of the same order of magnitude, and this may adversely affect RL. Our experiments demonstrated that scaling $\mathcal{L}_{reuse}$ to an order of magnitude with $\mathcal{L}_Q$ is a prudent solution to the problem. Algorithm 1 depicts the KnowRU algorithm based on the AC framework.

Algorithm 1: The training process based on AC framework.

Initialization: Parameters θ_A and θ_C represent student's actor and critic networks, respectively; parameter θ_P represents the previous policy model; and parameters α and β represent the weight of knowledge reuse and weight decay value, respectively.
Output: Parameter θ_A of the actor networks and parameter θ_C of the critic networks for every agent:
for episode = 1 to max-episodes:
  for step i = 1 to max-steps-in-episode:
    take action $a_i : a_i = \pi_{\theta_A}(s_i) + \mathcal{N}_i$
    get r_i from environment, and observes new state s_{i+1}
    store transition (s_i, a_i, r_i, s_{i+1}) into replay buffer
  end for
  randomly sample N transitions from replay buffer
  for j = 1 to N by step k:
    get n_j samples by order
    get $logits_P$ by $\pi_{\theta_P}(s_i)$
    get $logits_S$ by $\pi_{\theta_A}(s_i)$
    compute $\mathcal{L}_{reuse} = \frac{1}{n_j}\sum_i(\mathcal{L}_{function}(logits_P, logits_S))$
    where $\mathcal{L}_{function} := \mathcal{L}_{MSE}$
    get $\mathcal{L}_Q = \frac{1}{n_j}\sum_i(Q_{\theta_C}(s_i, a_i))$
    optimize actor network by minimizing: $\mathcal{L}_{all} = \alpha\mathcal{L}_{reuse} + (1-\alpha)\mathcal{L}_Q$
    optimise critic network by minimizing: $\mathcal{L}_{critic} = \frac{1}{n_j}\sum_i |(Q_{\theta_C}(s_i, a_i) - r_i)|$
  end for
  if $\alpha > 0.02$:
    $\alpha = \alpha - \beta$
end for

4. Experiments and Analysis

4.1. Experimental Setup

Multi-agent particle environment. We constructed three scenarios in the multi-agent particle environment (MPE) [14] to validate the performance of our method. The environment supports both cooperation and competition scenarios, and allows for the modification of existing scenarios. We tested KnowRU in both cooperation and competition scenarios based on the MADDPG and MAAC.

As mentioned earlier, knowledge reuse plays a vital role in continuous variational tasks. Therefore, we constructed a series of environments to simulate the variations in such tasks by changing the number of agents and landmarks. In our experiments, KnowRU had a positive impact on accelerating training and improving performance, primarily by comparing the results of basic algorithms and KnowRU. All the contrasts were based on the same experimental conditions. The experiments consisted of three scenarios involving cooperation and competition: simple_spread, simple_adversary, and cooperative_treasure_collection.

Performance Metrics. There is also a common standard to measure the success of knowledge reuse in our experiments, summarized by [16] and resumed by [5], as illustrated in Figure 3. The three main indicators are as follows: (1) **jump start**—measuring performance improvement at the beginning of training; (2) **time to threshold**—for tasks which may yield a significant result at some point, the learning time required to reach the level is meaningful; and (3) **asymptotic performance**—in complex tasks, agents may fail to achieve optimal performance and instead achieve a suboptimal one. Knowledge reusing may help agents in achieving higher performance, and the before-and-after performance gap is called asymptotic performance.

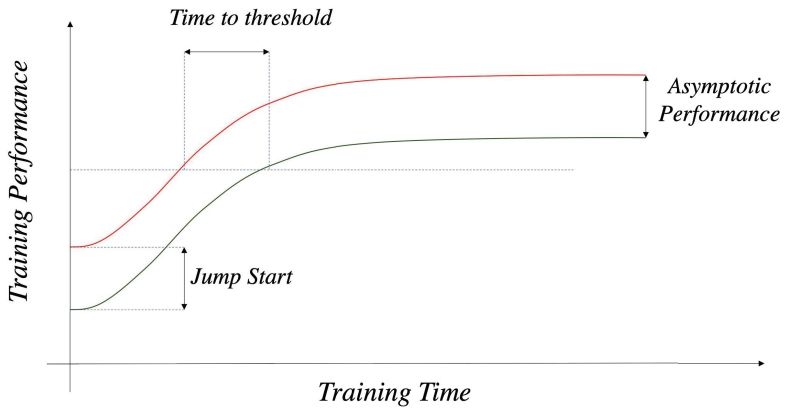


Figure 3. Transfer performance metrics.

4.2. Simple _Spread Scenario

A simple _spread is a predefined typical cooperative scenario of an MPE. In this environment, agents must cooperate to reach a set of landmarks (targets) without communication. The targets are not different, and agents must collect the shared rewards by arriving at all targets as quickly as possible. Meanwhile, agents are penalized when they collide with one another.

In our experimental settings, we took the task containing four agents and four targets as the previous policy models' training scenario, called Task I. In a real-world scenario, the number of agents and targets usually changes with the continued variation of the training mission. For this reason, we designed two scenarios to serve as current agents' training scenarios. Task II was made up of six agents and six targets, while Task III was made up of eight agents and eight targets, as illustrated in Figure 4a. In simple_spread, we used MADDPG as the baseline algorithm and implemented KnowRU based on it. In addition, we specially added a control group that initialized the current agent with the previous policy model, which is called "MADDPG with initialization", to verify the infeasibility of training directly based on the policy model.

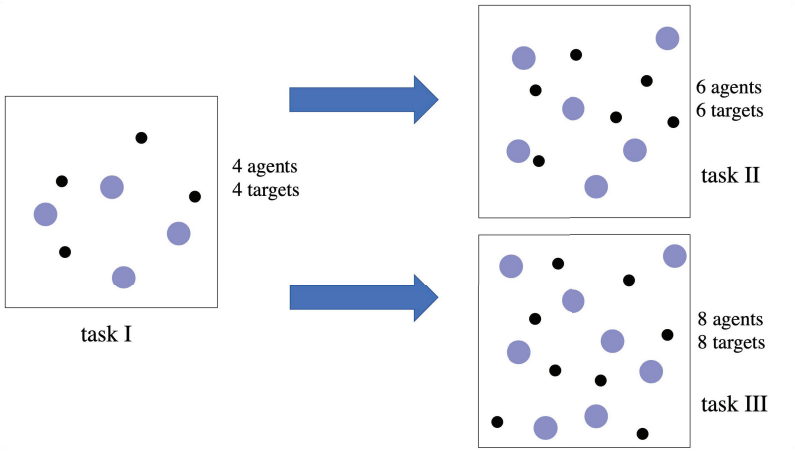
4.3. Simple _Adversary Scenario

A simple _adversary is a predefined typical competitive MPE scenario. In this scenario, n agents must work together to reach one target landmark from the total n landmarks (targets) without communicating. The agents must maximize the shared rewards by minimizing the distance between the right target and the nearest agent. Meanwhile, the adversaries also want to reach the target without knowing which one is right, and the agents are also penalized by the adversary's distance from the target. Because agents know the correct target point, the agents have an advantage at the beginning. As the training progressed, adversaries learned how to distinguish the correct target and achieve a balance of power.

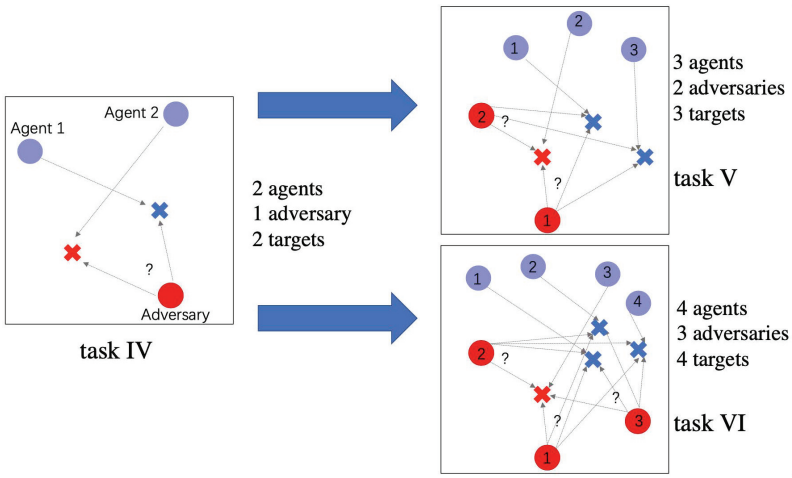
In our experimental settings, we took the task that contained two agents, one adversary, and two targets as the previous policy models' training scenario, called Task IV. We designed two scenarios as the current agents' training scenarios. Task V was made up of three agents, two adversary and three targets, while Task VI was made up of three agents, two adversary and three targets as illustrated in Figure 4b. For this scenario, we only implemented KnowRU with adversaries to help them distinguish the correct target at the beginning.

In the first two scenarios, we took the average step reward from the environments in each episode to measure the effect of training. The higher the reward, the better was the reward. The actor and critic networks in the experiments were all randomly parameterized by a four-layer fully connected MLP with 64 units per layer. The architecture of MLP is not strictly limited, only that the output layers of historical models and current models (that

is, the neural layer's output that determines the action) have the same action dimensions. We set 5000 episodes to ensure convergence, every episode consisted of 25 steps, and the networks were updated every four episodes. The hyperparameters α and i are set to 0.5 and 0.02, respectively. All the results and 95% confidence intervals (95% CI) are illustrated in Figures 5 and 6. The red lines represent the best performances in these scenarios.



(**a**) Tasks in simple_spread scenario.

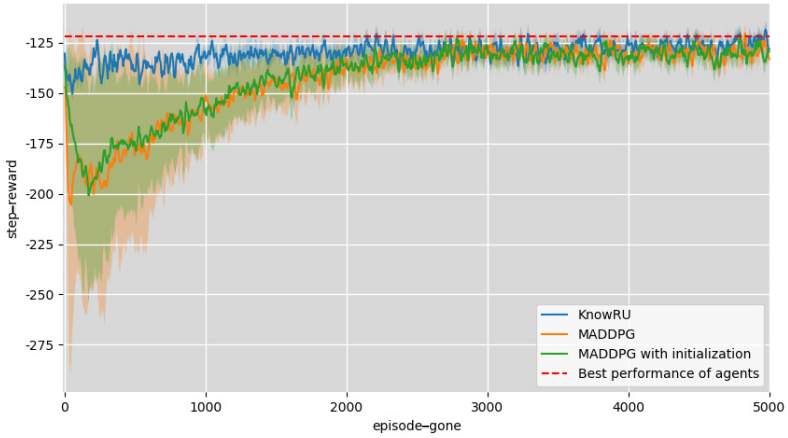


(**b**) Tasks in simple_adversary scenario.

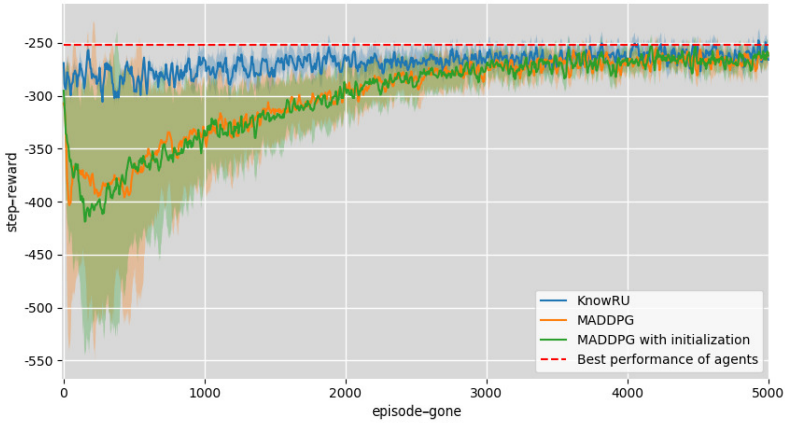
Figure 4. The tasks in Simple_spread & Simple_adversary.

Discussion. From Figures 5 and 6, it is obvious that the performance of KnowRU in all experiments was greatly improved at the beginning of training, and the episodes required for training convergence were also greatly reduced, satisfying two of the three indicators, and demonstrating that KnowRU successfully reuses knowledge. It is worth mentioning that when we trained the initialized agents in Tasks II and III, compared to MADDPG, the results did not clearly improve. This was caused by the characteristics of the models, which usually only work well in the scenarios where they are trained. When the task changes, the former models are no longer applicable because of over-fitting. Meanwhile, MADDPG showed considerable fluctuations in the training process without prior knowledge, particularly in the early stages of training, and performance decreased significantly. As for KnowRU, the convergence results were almost obtained at the beginning, achieving the goals of reusing knowledge in new tasks. Compared to MADDPG, KnowRU showed excellent performance and small fluctuations during training. We believe that the reason KnowRU works is that it effectively narrows the solution space by providing more prior knowledge, thereby narrowing the space for exploration. We also found that during the training process, the fluctuation of the loss function value

caused an unstable performance of the agents. When we used KnowRU to train the agents, the oscillating amplitude of the loss value for backpropagation was apparently smaller, and $\mathcal{L}_{reuse}$ first decreased and then increased, demonstrating the effectiveness of KnowRU's two-phrase design. KnowRU does not appear to learn beyond its initial performance because agents achieved the best performance in the scenarios.



(**a**) task II in Simple_spread.

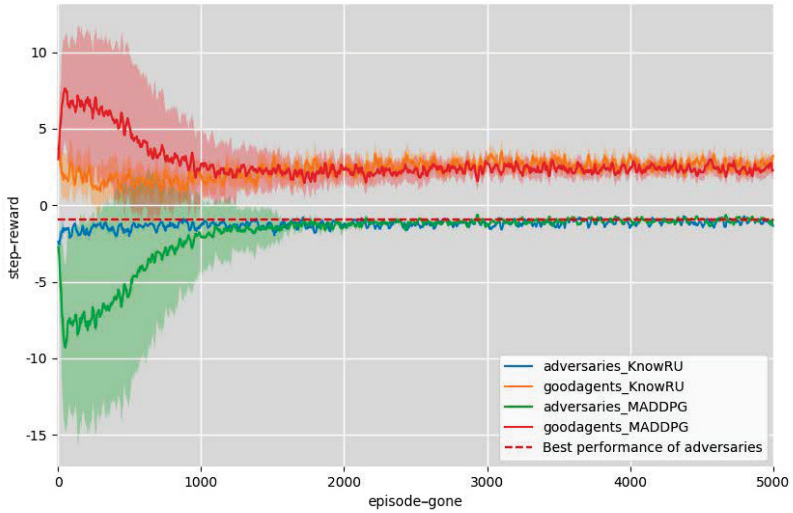


(**b**) task III in Simple_spread.

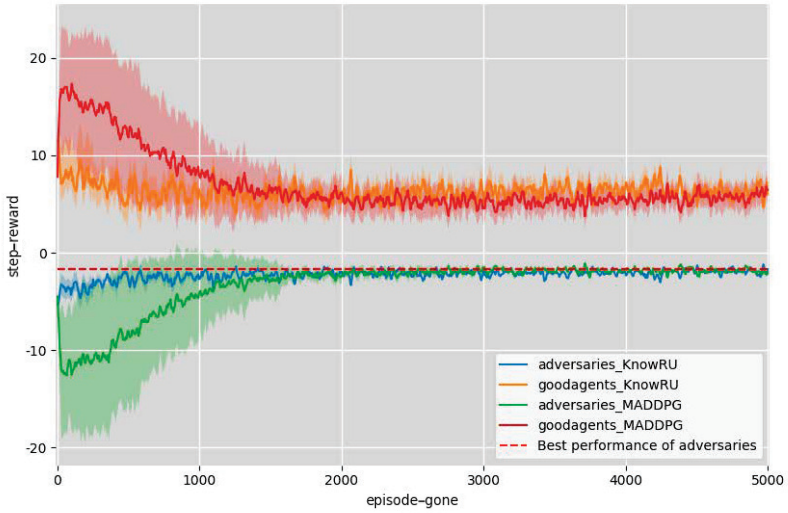
Figure 5. Knowledge reusing in Simple_spread.

4.4. Cooperative _Treasure _Collection Scenario

Cooperative_treasure_collection is a more complex competitive scenario constructed by [15] based on the framework of the MPE. In this scenario, there are two types of agents: X treasure collectors and Y treasure banks. There are also X treasures that correspond to the color of the bank. The collector's role is to collect the treasures of any color and transport them to the bank of the corresponding color. The treasures are reborn after being collected, and the banks simply gather as many treasures as possible from collectors. When the treasures are collected by the collectors, the collectors will share a global reward.



(**a**) task V in Simple_adversary.



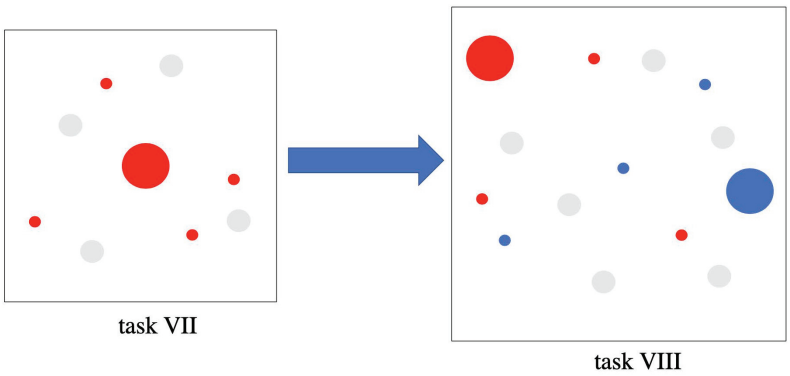
(**b**) task VI in Simple_adversary.

Figure 6. Knowledge reusing in Simple_adversary.

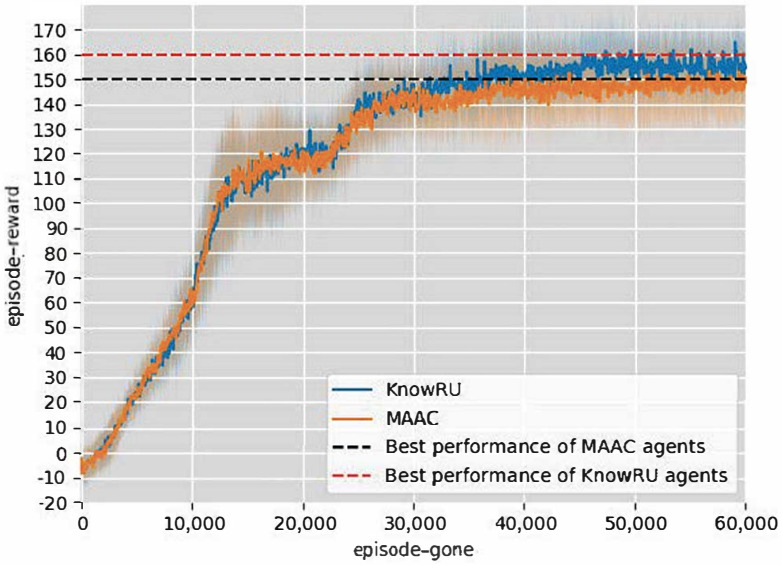
At the same time, while the treasure is transported into the bank, all agents receive a global reward. However, collectors are penalized if they collide with one another. To conclude, the collectors must learn how to collect treasures cooperatively and deposit them into the correct color bank as quickly as possible without colliding with other agents. Banks need to cooperate with collectors to place treasures.

In our experimental settings, we took the task which contained four collectors, one bank, and four treasures as the teacher models' training scenario, called Task VII. We designed a scenario which consists of six collectors, two banks, and six treasures as student model training scenarios, called Task VIII, as shown in Figure 7a. To test the universality, we used MAAC as the basic algorithm, which can learn these tasks well using the attention mechanism and implement KnowRU based on it. The actor networks in our experiments were randomly parameterized by a four-layer fully connected MLP with 64 units per layer, and critic networks were parameterized by the attention mechanism. We set 60,000 episodes to ensure convergence; each episode consisted of 100 steps, and the networks were updated four times every episode. The hyperparameters α and i were set to 0.5 and 0.02, respectively. We took the average episode reward of all agents from the environment to measure the effect of training. The higher the reward, the better. The results and 95% confidence intervals (95% CI) are illustrated in Figure 7b.

Discussion. As shown in Figure 7b, we completed the time to threshold and asymptotic performance goals. MAAC first obtained and stabilized a reward value of 150 at approximately 37,000 episodes. However, KnowRU first achieved approximately 29,000 episodes and advanced this time by approximately 21.6% compared to MAAC. In comparison to the other two scenarios, we can draw different conclusions and conjectures from this scenario. Even though the two tasks in this scenario are not closely related, KnowRU also successfully helps agents in avoiding the local optima and achieving higher performance. The result proves our conjecture that KnowRU can help and guide the training of agents by providing more useful information.



(**a**) The tasks in cooperative_treasure_collection scenario.



(**b**) Knowledge reusing in task VIII.

Figure 7. The tasks and results in Cooperative_treasure_collection.

4.5. Component Analysis and Discussion

Alpha. We have declared that annealing of the alpha parameter blends the guide phase and specialization phase. We intended to explore the impact of alpha on performance using Equations (10) and (11).

$$T = \{T_{\alpha 1}, T_{\alpha 2}, T_{\alpha 3} \cdots T_{\alpha n}\} \tag{10}$$

$$\text{Performance}_{\alpha} = \ln\left(\frac{\text{T}_{\max}}{T_{\alpha}}\right) \tag{11}$$

where T_α represents the moment when the best performance is achieved with α, T is the set of T_α, $T_{\max}$ is the maximum value of the set T, and performance $_\alpha$ denotes the performance of α. We tested the performance in Task III of the simple_spread scenario, and the results are shown in Figure 8.

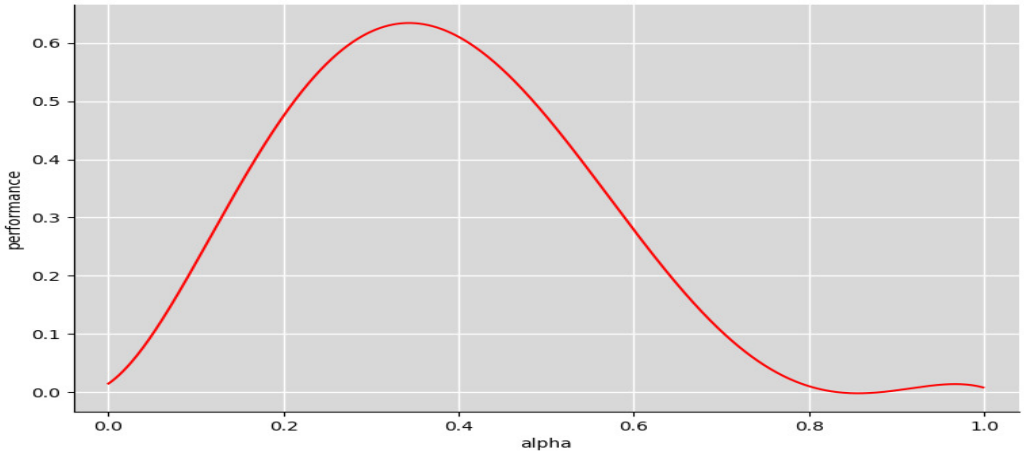


Figure 8. The impact of alpha on the moment when the best performance is achieved.

When alpha is greater than 0.1, our method clearly outperforms the baseline. Here, we explored the best settings for alpha. When the alpha value is approximately 0.3∼0.5, the training performance is optimal, which is in line with our expectation that the training process is first guided and then specialized. We also found similar patterns in the other tasks.

Loss Function. There are three viable loss functions for $\mathcal{L}_{reuse}$ tested: MSE loss, CE loss, and KL loss. The logits can be converted into probabilities with a softmax function, which can then be used in CE or KL. We found that using different loss functions had no discernible effect on the experimental results.

Discussion. We tried different combinations of the components. The results did not show a significant difference, and they all demonstrated that KnowRU successfully helped agents in quickly adapting to the environment, proving the effectiveness and robustness of our method.

5. Conclusions

The transferring of knowledge from historical experiences is of great practical importance in the MARL fields; however, it is notoriously unstable and difficult. In this study, we explored a novel knowledge transfer approach for MARL and addressed its accompanying unique challenges, leveraging the KD paradigm. To empirically demonstrate the robustness and effectiveness of KnowRU, we performed extensive experiments on state-of-the-art MARL algorithms in collaborative and competitive scenarios. The results demonstrate the effectiveness and robustness of KnowRU under different experimental settings.

Author Contributions: Conceptualization, methodology, software, investigation, Z.G. and K.X.; resources, data curation, writing—review and editing, project administration, funding acquisition, funding acquisition, B.D. and H.W.; writing—original draft preparation, Z.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work is partially supported by the major Science and Technology Innovation 2030 "New Generation Artificial Intelligence" project 2020AAA0104803 and the National Natural Science Foundation of China (No. 61751208).

Data Availability Statement: The source code of KnowRU is published on: https://github.com/KnowRU-MARL/KnowRU.

Conflicts of Interest: The authors declare no conflict of interest. We wish to confirm that there are no known conflicts of interest associated with this publication and there has been no significant financial support for this work that could have influenced its outcome. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
2. Tesauro, G. Temporal difference learning and TD-Gammon. *Commun. ACM* **1995**, *38*, 58–68. [CrossRef]
3. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
4. Zhou, X.; Wang, H.; Ding, B. How many robots are enough: a multi-objective genetic algorithm for the single-objective time-limited complete coverage problem. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–26 May 2018; pp. 2380–2387.
5. Da Silva, F.L.; Costa, A.H.R. A survey on transfer learning for multiagent reinforcement learning systems. *J. Artif. Intell. Res.* **2019**, *64*, 645–703. [CrossRef]
6. Crandall, J.W. Just add Pepper: extending learning algorithms for repeated matrix games to repeated markov games. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, Valencia, Spain, 4 June 2012; pp. 399–406.
7. Hernandez-Leal, P.; Kaisers, M. Towards a fast detection of opponents in repeated stochastic games. In *International Conference on Autonomous Agents and Multiagent Systems*; Springer: Cham, Switzerland; 2017; pp. 239–257.
8. Kelly, S.; Heywood, M.I. Knowledge transfer from keepaway soccer to half-field offense through program symbiosis: Building simple programs for a complex task. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 1143–1150.
9. Hinton, G.; Vinyals, O.; Dean, J. Distilling the knowledge in a neural network. *arXiv* **2015**, arXiv:1503.02531.
10. Bowling, M.; Veloso, M. *An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning*; Technical Report; Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science: Pittsburgh, PA, USA, 2000.
11. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
12. Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. *Deterministic Policy Gradient Algorithms*; International Conference on Machine Learning; PMLR: Beijing, China; 2014; pp. 387–395.
13. Konda, V.R.; Tsitsiklis, J.N. Actor-critic algorithms. In *Advances in Neural Information Processing Systems*; MIT Press: Denver, CO, USA, 2000; pp. 1008–1014.
14. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*; MIT Press, Los Angeles, CA, USA; 2017; pp. 6379–6390.
15. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning (PMLR), Long Beach, CA, USA, 9–15 June 2019; pp. 2961–2970.
16. Taylor, M.E.; Stone, P. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.* **2009**, *10*, 1633–1685.
17. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 July 1993; pp. 330–337.
18. Han, J.; Hu, R. Deep fictitious play for finding Markovian Nash equilibrium in multi-agent games. In Proceedings of the Mathematical and Scientific Machine Learning (PMLR), Princeton, NJ, USA, 20–24 July 2020; pp. 221–245.
19. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296.
20. Zhou, M.; Chen, Y.; Wen, Y.; Yang, Y.; Su, Y.; Zhang, W.; Zhang, D.; Wang, J. Factorized q-learning for large-scale multi-agent systems. In Proceedings of the First International Conference on Distributed Artificial Intelligence, Beijing, China, 13–15 October 2019; pp. 1–7.
21. Koga, M.L.; Freire, V.; Costa, A.H. Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning. *IEEE Trans. Cybern.* **2014**, *45*, 77–88. [CrossRef] [PubMed]
22. Didi, S.; Nitschke, G. Multi-agent behavior-based policy transfer. In Proceedings of the European Conference on the Applications of Evolutionary Computation, Porto, Portugal, 30 March–1 April 2016; pp. 181–197.
23. Buciluǎ, C.; Caruana, R.; Niculescu-Mizil, A. Model compression. In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery And Data Mining, Philadelphia, PA, USA, 20–23 August 2006; pp. 535–541.
24. Lai, K.H.; Zha, D.; Li, Y.; Hu, X. Dual policy distillation. *arXiv* **2020**, arXiv:2006.04061.
25. Wadhwania, S.; Kim, D.K.; Omidshafiei, S.; How, J.P. Policy distillation and value matching in multiagent reinforcement learning. *arXiv* **2019**, arXiv:1903.06592.

26. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
27. Pan, S.J.; Yang, Q. A survey on transfer learning. *IEEE Trans. Knowl. Data Eng.* **2009**, *22*, 1345–1359. [CrossRef]
28. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 9. [CrossRef]
29. Kukačka, J.; Golkov, V.; Cremers, D. Regularization for deep learning: A taxonomy. *arXiv* **2017**, arXiv:1710.10686.