applied sciences

MDPI

Article

Message Action Adapter Framework in Multi-Agent Reinforcement Learning

Bumjin Park [1] **and Jaesik Choi** [1,2,*]

[1] KAIST AI, Daejeon 34141, Republic of Korea; bumjin@kaist.ac.kr
[2] INEEJI, Seongnam-si 13558, Republic of Korea
* Correspondence: jaesik.choi@kaist.ac.kr

Abstract: Multi-agent reinforcement learning (MARL) has demonstrated significant potential in enabling cooperative agents. The communication protocol, which is responsible for message exchange between agents, is crucial in cooperation. However, communicative MARL systems still face challenges due to the noisy messages in complex multi-agent decision processes. This issue often stems from the entangled representation of observations and messages in policy networks. To address this, we propose the Message Action Adapter Framework (MAAF), which first trains individual agents without message inputs and then adapts a residual action based on message components. This separation isolates the effect of messages on action inference. We explore how training the MAAF framework with model-agnostic message types and varying optimization strategies influences adaptation performance. The experimental results indicate that MAAF achieves competitive performance across multiple baselines despite utilizing only half of the available communication, and shows an average improvement of 7.6% over the full attention-based communication approach. Additional findings suggest that different message types result in significant performance variations, emphasizing the importance of environment-specific message types. We demonstrate how the proposed architecture separates communication channels, effectively isolating message contributions.

Keywords: multi-agent reinforcement learning; task adaptation; transfer learning

check for updates

Citation: Park, B.; Choi, J. Message Action Adapter Framework in Multi-Agent Reinforcement Learning. *Appl. Sci.* **2024**, *14*, 10079. https://doi.org/10.3390/app142110079

Received: 5 October 2024
Revised: 28 October 2024
Accepted: 30 October 2024
Published: 4 November 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Artificial intelligence (AI) has shown superior performance to humans, affecting several research fields. RL agents surpass human-level control in games such as Go and Dota [1,2]. The social influence of MARL will be increased [3]. Recent progress in deep neural networks shows the potential for artificial multi-agents [4,5]. In a shared environment, multiple agents are jointly trained to maximize a common objective by communicating their observations or states. With cooperation, agents can solve complex problems that single or non-cooperative agents can not solve [6]. In addition, several benefits exist with cooperation, including reduced individual workloads of individual agents [7]. Such benefits are leveraged in various applications, including power control, IoT (Internet of Things), and robotics [8]. Under the communication of several agents, verification of the effects of communication is crucial.

Despite the benefits of cooperation, the training dynamics of MARL show severe instability [9,10]. Among several reasons, such as the hardness of an opponent modeling and extensive search spaces of action, one crucial factor is the communication channels in MARL, which are optimized jointly with a policy to maximize rewards. In the learning dynamics of message construction, agents must learn to process observations to construct messages for other agents while interpreting received messages. Most MARL approaches utilize deep neural networks for the communication channels. However, optimizing message representation has been shown to increase the instability [11,12].

The instability is not only an issue for MARL but is already known for a single agent. For example, a nonstationary target problem exists in RL compared to the stationary target

in the computer vision classification task [13]. Accordingly, MARL, an extension of a single-agent RL, also includes the problem of the nonstationary target. Even worse, MARL utilizes noisy message representations for policy inputs. As such, it suffers from instability training and requires extensive training time. Although MARL could provide higher returns than individual training of agents, the contribution of MARL communication channels is not easily measurable, and this hinders the direct application of MARL, where verification of agent controls is crucial. For example, users can not deploy MARL agents in the real scenario of UGV (Unmanned Ground Vehicle) controls [14] without verifying the trained policies. To resolve the uncertainty on the contribution of communication channels in MARL, we factorize the actions conditionally on two variables: single-agent observation and messages from other agents. As such, we propose a message adapter framework (MAAF) where agents learn their actions without communication and then adapt the residual action conditional on the message while either updating or freezing the original action. This framework completely isolates the impact of messages from the observation-only control of an agent. The factorization of actions conditional on different inputs is highly important in control tasks where safe use of RL [15]. Consider a robot manipulation, where the individual robot moves following safe procedures. The disentanglement is barely achieved when a policy combines observations and messages as input.

MAAF can be interpreted as a transfer learning framework from individual control to communicative action problems. We define two types of actions, observation-action and message-action, and study the difference between the two actions. Figure 1 shows two action types derived from observations and messages in a predator–prey environment. The proposed framework explicitly infers these two types of actions.

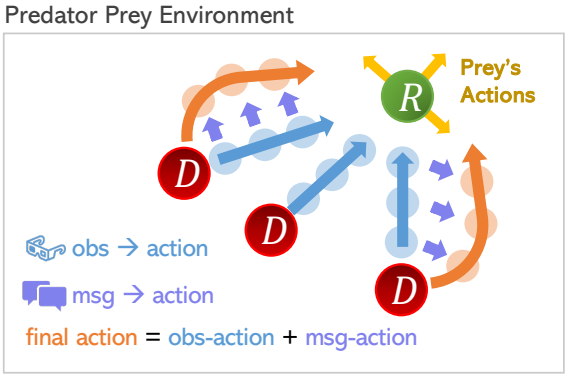


Figure 1. Graphical illustration of two types of actions. The observation action is a non-cooperative behavior in which the prey can easily escape the predators. On the other hand, the messages encourage better cooperative behavior. The messages contribute to residuals of actions; the final actions combine the base actions and the residuals.

This work studies the separation of messages in MARL to encourage factorized modeling of MARL action space. We propose message-based action adaptation (MAAF) to verify the contribution of message factors. We evaluate MAAF on four multi-agent environments and analyze the contribution of message factors. We also provide optimization strategies for MAAF and counterfactual analysis with a counterfactual conditional statement for a non-communicative agent, "What if an agent receives messages?". We summarize our contributions as follows:

- We propose MAAF, which decouples the training of cooperative controls. The experimental results show that MAAF achieves competitive performance to full communication-based learning while using only half the communication.
- We leverage the adaptation framework to evaluate combinatorial inputs for message construction and experimentally demonstrate that message types significantly impact performance.

2. Related Work

2.1. Multi-Agent Reinforcement Learning

MARL is a natural extension of RL for multi-agents. Several RL algorithms are utilized in MARL, such as off-policy DQN (Deep Q-Network) [16] and SAC (Soft Actor–Critic) [17], and on-policy algorithms, A2C (Asynchronous Actor–Critic) [18] and PPO (Proximal Policy Optimization) [19]. Unlike a single-agent RL, whose Markov decision process (MDP) assumes a transition of states with a single-agent action, MARL evolves actions of multiple actors that increase the complexity of the environment dynamics. To efficiently train MARL, a notable approach is a shared Q-value function to reduce value function errors. MADDPG (Multi-Agent Deep Deterministic Policy Gradient) [20] estimates Q-values for individual agents conditional on the observations and actions sampled by individual agents. QMIX (Q-Mixing) [21] combines Q-values of individual agents, modeling monotonic increasing function with non-negative weight. This work uses PPO as a base policy for a single agent.

MARL settings could be centralized or decentralized by the information sharing principle between multi-agents [4]. Most works follow centralized training and decentralized execution (CTDE), as most MARL environments can not acquire full communication between agents in the execution phase. This work assumes a decentralized setting with message communications. When no messages are transmitted, our setting is fully decentralized [22]. In other words, only messages contribute to the MARL, and no shared observations are used to estimate values.

More recent discussion on MARL includes a comparison of on-policy and off-policy in MARL [23], the emergence of cooperative behavior [24], stability of training [9], modeling social influence [25], and architectural modification of controls [26]. Theoretically, [27] propose HAPPO (Heterogeneous-Agent Proximal Policy Optimisation) with the multi-agent advantage decomposition theorem, which requires sequential modeling of MARL actions, and another work further utilized by transformer architecture [26]. In the rapid progress of MARL modelings, this work studies the message component in MARL for a better interpretation of multiple agents.

2.2. Task Adaptation in MARL

This work is closely related to adaptation as we separate the MARL training phase into (1) single-agent training and (2) message adaptation training with two policies conditional on observation and message, respectively. Adaptation of agents is crucial for generalization perspectives, including domain adaptation [28,29], continual learning [30], and meta-learning [31]. Domain adaptation studies the change in actions conditional on the context vectors of an environment. PEARL (Probabilistic Embeddings for Actor–critic RL) [32] uses the Bayesian perspective on context inference; CaDM (Context-aware Dynamics Model) [33] trains an agent to predict forward and backward dynamics. These works use conditional vectors to transform an action into a task-specific action. Recently, an action translator for single RL domain adaptation has been proposed [34]. The action translator modifies a current action into a domain-specific action conditional on the context of an environment. Similarly, we isolate the message component by modeling an action translator with a residual stream. The residual term stabilizes large models, and several modern neural networks utilize the residual stream [35]. In RL, the model-based approach for continual action [36] learns a residual to model updated actions. This work also utilizes residuals of the action derived from a message to interpret two types of actions. We formulate our problem as a transfer learning from single agents to communicative agents.

2.3. Communication in MARL

Modeling communication with neural networks is essential to make robust, efficient MARL. Several works model social intersection in MARL to improve MARL algorithms. Communication components are either explicit, like language and discrete symbols, or implicit, like differential communication. Differential communication makes inconsistent messages unlike explicit components, also called *common grounding*. Ref. Guan et al. [37] utilizes an

autoencoder to make a compact representation from pixels to shape the common grounding. Ref. Jaques et al. [25] models causal inference, which maximizes the mutual information between agent actions. Another way to model intersection is opponent modeling. Ref. Wu et al. [38] models the Bayesian inference on the opponent modeling. A recent work utilizes large language models to learn intersections in social settings [39]. Another research branch studies neural network modules for communication. TarMAC (Targeted Multi-Agent Communication) [11] models attention-based communication. I2C (Individually Inferred Communication) [40] further improves communication by limiting communication channels. CommFormer [41] introduces a learnable communication graph to optimize inter-agent coordination, allowing agents to update communication pathways through bi-level optimization dynamically. IMGS-MAC (Information Maximizing Gated Sparse Multi-Agent Communication) [42] applies an information bottleneck approach to achieve sparse communication, enabling agents to communicate effectively under limited bandwidth constraints. CAMA (Complementary Attention for Multi-Agent reinforcement learning) [43] uses complementary attention mechanisms to enhance agent cooperation in variable team sizes, addressing issues of distracted attention. Although these works shed light on message flow, the communication channels are highly entangled with observations, making it difficult to verify the exact contribution of messages in determining actions. We solve this problem by modeling explicit action branches for observation and messages.

2.4. Noisy Inputs for Neural Networks

Under the non-stationary issues in MARL, the differential messages further distract the stabilization. Recent work [13] shows that the internal neurons become dormant by the non-stationary target. In addition, several works show that simple MLP (Multi-Layer Perceptron) performs better than complex RNN (recurrent neural network) structures [44,45]. Although enormous resources, including training time and hyperparameter tuning, can find a sub-optimal policy forming good differential messages, even a tiny noise on message representation can be problematic as neural networks are usually non-robust and weak at adversarial attacks [46]. Therefore, this work assumes that differential messages are non-robust features, and another module must separate the impact of messages.

3. Problem Setting

RL is a Markov decision process $\mathcal{M}(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ with a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ which returns the probability of the next states, and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$, and finally, a discount factor $\gamma \in [0, 1]$. The general MARL setting can be written as MDP

$$\mathcal{M}(N, \mathcal{S}, \{\mathcal{A}^i\}_{i\in\{1,\cdots,N\}}, P, \{\mathcal{R}^i\}_{i\in\{1,\cdots,N\}}, \gamma) \tag{1}$$

where N is the number of agents, $\mathcal{A}^i$ and $\mathcal{R}^i$ are action space and reward function for agent i individually [47]. Here, transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A}^i \times \cdots \times \mathcal{A}^N \to \Delta(\mathcal{S})$ is defined for the joint actions of multi-agents. In the case of communicative multi-agents, we add message space and optionally, a partially observable condition for a POMDP case. In the first step of MAAF, we train an agent's policy based on the observation $o \in \mathcal{O}^i$ to maximize the rewards. Then, the agent learns to communicate with messages as inputs. We consider it as the change in MDP from a base MDP $\mathcal{M}_B$ to another MDP $\mathcal{M}_T$ as follows

$$\begin{aligned} &\mathcal{M}_B(N, \mathcal{S}, P, \{\mathcal{O}^i\}_{[N]}, \{\mathcal{A}^i\}_{\{[N]}, \{\mathcal{R}^i\}_{[N]}, \gamma) \Rightarrow \\ &\mathcal{M}_T(N, \mathcal{S}, P, \{\mathcal{O}^i\}_{[N]}, \{\mathcal{H}^i\}_{[N]}, \{\mathcal{A}^i\}_{[N]}, \{\mathcal{R}^i\}_{\{[N]}, \gamma) \end{aligned} \tag{2}$$

where $\mathcal{H}^i$ is the message space obtained for agent i and $\{\cdot\}_{[N]}$ is the simplification of notation $\{\cdot\}_{i\in[N]}$. This transformation of MDP can be viewed as a transfer learning framework, and we change only the input feature to include cooperative messages while the remaining conditions are fixed.

The transfer learning formulation assumes only the inclusion of message features $\{\mathcal{H}^i\}_{[N]}$, and the learning strategy in both MDPs $\mathcal{M}_B$ and $\mathcal{M}_T$ could be any MARL algorithms such as MADDPG [20] and QMIX [21]. Note that some MARL algorithms use shared Q-values, serving as implicit communication to improve Q-value estimation accuracy. Since this work focuses on verifying the effects of communication channels, we do not incorporate value sharing in training the proposed algorithm.

4. Methods

This section describes the MAAF, which has two action branches to compute action logits independently. Previous policy networks combine messages with observations and infer actions. On the other hand, the proposed policy network integrates two types of action logits. Figure 2 compares two approaches. The proposed network is based on two key modeling assumptions in communication. We discuss two assumptions from a single-agent perspective and describe the MAAF from the MARL perspective.

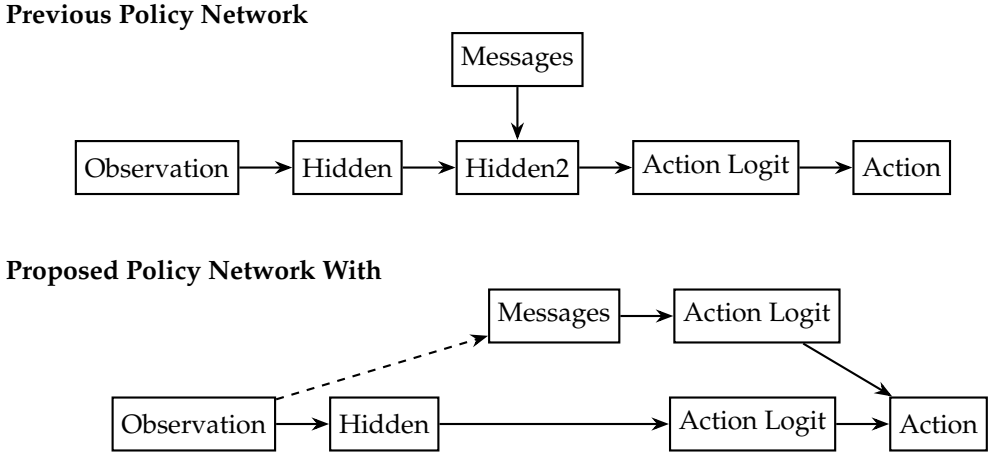


Figure 2. Graphical models for the forward action process in policy networks. Previous MARL methods link hidden representations to an intermediate hidden layer (Hidden2), resulting in entangled representations. In contrast, the proposed action adaptation restricts the contribution of messages to the action logit, offering a more disentangled integration. The interaction between observation and messages can be constructed by passing observation to the message (dashed line).

4.1. Modeling Assumptions

MAAF is based on the following assumptions in communication learning.

Assumption 1 (Minimal Interference)**.** *Actions determined by observation information and message information exhibit minimal interference.*

Assumption 2 (Efficient Independent Message Adaptation)**.** *In inter-agent communication learning, training starts from a pre-trained state without prior communication.*

Assumption 1, minimal interference, says the two types of predicted actions contribute to the action with minimal interference. This can be achieved through a linear combination of actions. For discrete actions, this is implemented by adding two types of action logits, while for continuous actions, it involves adding two continuous actions. Assumption 2, efficient and independent message adaptation, suggests that communication learning begins from a trained state for efficiency and is independent of prior communication, allowing verification of communication contribution from no communication state. This work implements Assumption 2 by pre-training agents without communication and adapting communication by adding message–action modules at the action logits. The interaction between an observation feature and messages (from other agents) is encoded in the message part, as illustrated in the dashed line in Figure 2. Therefore, the proposed framework

has similar expressive power with the previous feature encoding, which concatenates the observation features and the messages. The benefit of this modeling is the separation of (a) a complex decision derived from multi-agent messages and (b) a relatively simple decision derived from single-agent observation.

4.2. Message Action Adapter Framework

Figure 3 illustrates MAAF for single-agent and multi-agent perspectives. MAAF has two branches, base policy, and adapter policy, that compute the final multi-agent actions. We divide the overall training steps into two steps to train policies. The first step is to train individual actions based on their observations without communication. The second step is to make cooperative messages and update the base action with residual action. In detail, the base action A in the first step is updated to $\tilde{A}$ by additive operation $\tilde{A} = A + \hat{A}$ where $\hat{A}$ is the residual action obtained by the message sharing. In the update, the update of the base policy is an optimization choice, as keeping the single agent's action can provide interpretation and offset for the adapted actions. This simple separation of action generation steps can verify the importance of message features by analyzing two actions, base action A and adapter action $\tilde{A}$. Any information from the computation of the base action can be utilized to generate messages. We further discuss the input types of messages in Section 5.

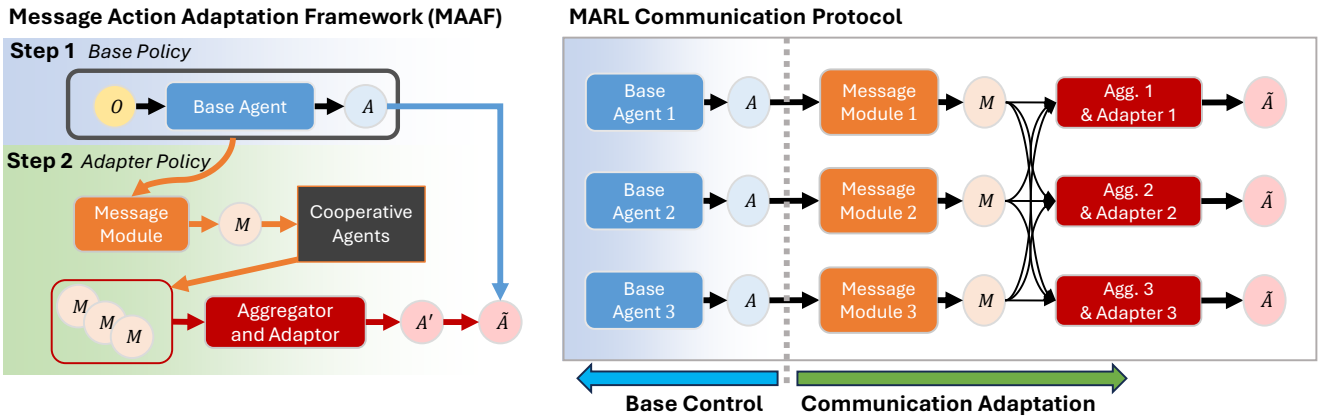


Figure 3. Graphical illustration of MAAF. (**Left**) Two steps for training: a base policy and a message adapter policy (blue and orange regions). In step 1, the agent learns to maximize the environment return without communication. In step 2, the agent generates messages and communicates with cooperative agents. The adapter policy produces a residual action A' resulting in the final action $\tilde{A}$. In the training of the adapter policy, the parameters of the base policy are fixed. (**Right**) Overview of MARL control for two steps. Each agent produces action and communicates by generating messages.

We formulate MAAF in detail. MAAF consists of two individual training steps in two MARL MDPs: single-agent training and multi-agent message adaptation. In single-agent training, the agents train base policies to maximize the rewards without communicating with each other. In the second step of the environment, the agents learn to adapt the base policies to utilize communication with the cooperative agents.

4.2.1. Step 1: Single-Agent Training

We train a single agent without any communication in the first step. For agent i, we train an agent policy π_i^{base} to maximize rewards that agent i obtains. The policy is conditional on the observation o_i of the agent and the policy π^{base} sample action

$$a_i^{base} \sim \pi_i^{base}(a_i|o_i). \tag{3}$$

The first step can be interpreted as base control training in which the agent achieves the supreme return without message communication. Although this work assumes the

same environment for steps 1 and 2, different environments can be considered, such as a single robot control and cooperative controls.

4.2.2. Step 2: Communication with Messages

After training the base policies for all agents, the agents communicate to leverage the observed environment information to construct messages. These messages are further applied to cooperative agents to make final actions. We train a message generator g_i, message interpreter f_i, and action adapter ϕ_i which will be combined to base policy network π_i^{base}. The message generator inputs any information for the agent to construct a message and outputs a vector $m \in \mathbb{R}^{d_{msg}}$. The number of cooperative agents of agent i is not fixed. To process these messages, a usual choice of aggregation function of messages is a recurrent neural network (RNN), but other pooling methods or attention mechanisms, such as TarMAC, can be applied. We use notation $\bar{m}_i$ to represent the aggregated messages agent i received.

The message generated by an agent also feeds into the agent. This process can be viewed as a self-talk. Finally, the action adapter ϕ_i inputs the current observation and the aggregated messages to make residual action logits

$$\log \hat{\phi}_i(a_i|o_i, \bar{m}_i) = \log \pi_i^{base}(a_i|o_i) + \log \phi_i(a_i|o_i, \bar{m}_i), \tag{4}$$

where $\hat{\phi}_i$ is a message-adapted policy of π_i^{base} (in short adapter policy). In training the adapter policy, the parameters of the base policy could be either updated or not, but the messages do not directly affect the base policy. We provide a pseudo-code of the action computation in Appendix A. Although we formulate a message adapter with a policy, the q-value estimator in an off-policy setting can be formulated similarly by adding a residual term of q-values. The adapted action in MAAF is

$$a_i^{adapt} \sim \hat{\phi}_i(a|o_i, \bar{m}_i) \tag{5}$$

and the contribution of the message can be quantified with a measure $d(\cdot, \cdot)$ as follows:

$$d\Big(\hat{\phi}_i(a_i|o_i, \bar{m}_i), \pi_i^{base}(a_i|o_i)\Big). \tag{6}$$

For example, when we choose KL divergence for the measure, the contribution of a message is the discrepancy between adapted policy $\hat{\phi}$ and adapter ϕ:

$$D_{KL}\Big(\hat{\phi}_i(a_i|o_i, \bar{m}_i)||\pi_i^{base}(a_i|o_i)\Big) \tag{7}$$

$$= \sum \hat{\phi}_i(a_i|o_i, \bar{m}_i) \log \frac{\hat{\phi}_i(a_i|o_i, \bar{m}_i)}{\pi_i^{base}(a_i|o_i)} \tag{8}$$

$$= \sum \hat{\phi}_i(a_i|o_i, \bar{m}_i) \log \phi_i(a_i|o_i, \bar{m}_i) \tag{9}$$

$$= \mathrm{H}\left(\hat{\phi}_i(a_i|o_i, \bar{m}_i), \phi_i(a_i|o_i, \bar{m}_i)\right) \tag{10}$$

where $\mathrm{H}(\cdot, \cdot)$ is the cross entropy of two probability distributions. Hence, the difference between the message-adapted and base policies can be written as the difference between the adapted policy and the residual part.

5. Message Types

We can utilize several message types for a fixed base policy and train different action adapters ϕ_i for message types to adapt differently. In particular, when inputs for message generation are noisy, we can detach the adapter and control it only with the base policy π^{base}. The idea that different messages help in different situations opens a problem of message input selection. Figure 4 shows the graphical illustration of their combination. We list four types of representations for message generation inputs.

- Observation (o): the raw observation of the agent.
- Action (a): the determined action of the base policy.
- Probabilistic distribution of actions (p): the preference of actions before sampling.
- User-defined message (optional): a user-defined message input.

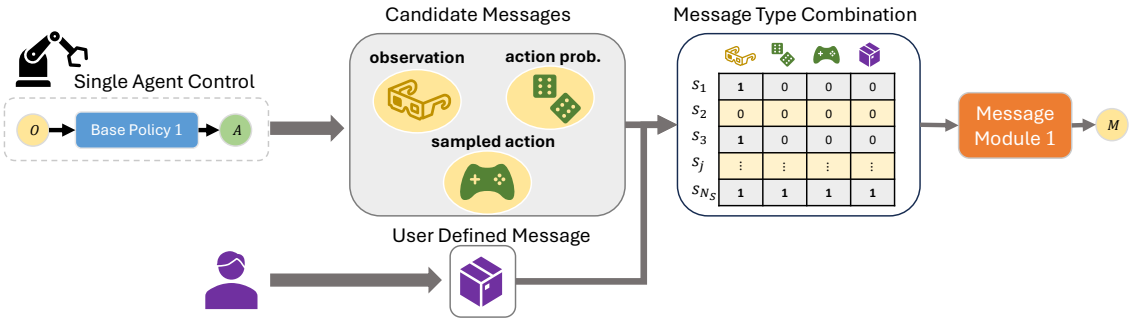


Figure 4. Graphical illustration of message generation. The combination of four types generates message M for cooperative agents. Optionally, a user can feed a new message input type to utilize communication.

The effectiveness of certain information for communication depends on the environment. For example, in the predator–prey environment, predators can efficiently surround the prey when they know the expected behavior of cooperators. Considering the instability and noisy representation message space in MARL, adding all the message inputs could result in lower performance. Therefore, we may need to search for the optimal combinations. In this work, the goal of searching combination is to compare the impact of the message components in MARL rather than finding the optimal policy. We may find the optimal policy by adding all the message inputs and training modules with lots of computational resources. Instead, this work uses an equal training time for all combinations and compares the contribution of message inputs.

6. Experiments

We evaluate the proposed MAAF by comparing three MARL algorithms: IPPO (Independent PPO) [23], which consists of independent PPO agents [19] without value sharing; MAPPO (Multi-Agent PPO) [23], which uses a shared value network by observation sharing; and TarMAC [11], which shares attention-based messages. We construct messages by adding a branch from the internal hidden representation to obtain the communication messages. The proposed methods have two variants for message aggregation: RNN-based and attention-based. For RNN, we use a two-layer GRU and a single-head for attention-based aggregation. The base policy PPO is implemented by CleanRL [48]. We train MARL agents for 1M time steps for base policy and another 1M for the adaptation. We use four types of environments: multi-particle environments (MPE) [49] the spread and the tag, and SMAC (Starcraft Multi-Agent Challenge) [50] 3M (3 Marines) and 8M. Figure 5 shows examples of environment states. Agents in the spread environment must spread as far as possible to each other while staying near target locations. Predator agents in the tag environment must tag as many prey as possible to achieve a higher return. As agents share rewards, reported MARL rewards are single-agent rewards. We train the baselines for three seeds and PPO clip coefficients (0.2, 0.3, 0.4). We report the best results obtained by the optimal clip coefficient. For MAAF, we train for three seeds and fix the clip coefficient of 0.3. We report the best message type in combinations $\mathcal{S}_{\text{comb}} = (o, a, a, p, op, ap, oap)$ where (o, a, p) represent observation, action, and probability distribution, respectively. The base policy network has 3 layers of 128 hidden dimensions with Tanh activation. The message generator is a 3-layer MLP with 128 hidden dimensions and Tanh activation. The dimension of generated messages is eight. The adapter policy is the same architecture as the base policy except that the input is an eight-dimensional hidden representation of messages.

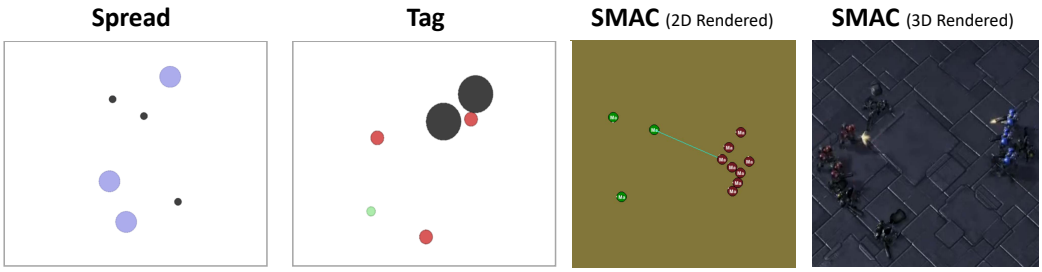


Figure 5. Rendering of three environments. In the spread environment, three agents (purple circles) must spread as far as possible while staying near target locations (small black circles). In the tag environments, three predators (red circles) chase prey (a green circle) in an environment with obstacles. In SMAC environments, two groups of agents fight each other.

7. Results

For each section, we answer the following questions:

1. Optimization Strategy: Is it better to freeze the base policy during the second step?
2. Performance of MAAF: Does MAAF outperform baseline models?
3. Effect of Message Types: How do different message types impact performance?

7.1. Optimization Strategy

MAAF has two separate steps in training: base policy training and message adaptation training. Note that this separation can encourage pre-trained RL agents to adapt to communication. When base policy parameters are frozen (stop-gradient and no parameters update), the original action distribution is preserved, and residuals are trained. This will result in a little discrepancy in actions between pre- and post-adaptation. Figure 6 shows the adaptation results for both cases with MAAF-RNN, including a scratch case where both base and adapted policies are trained from the beginning. We observe that the proposed method works to reach the performance of the scratch case when the parameters of base policies are updated. Note that freezing the base policy results in a little discrepancy between the performance at 1M and 2M.

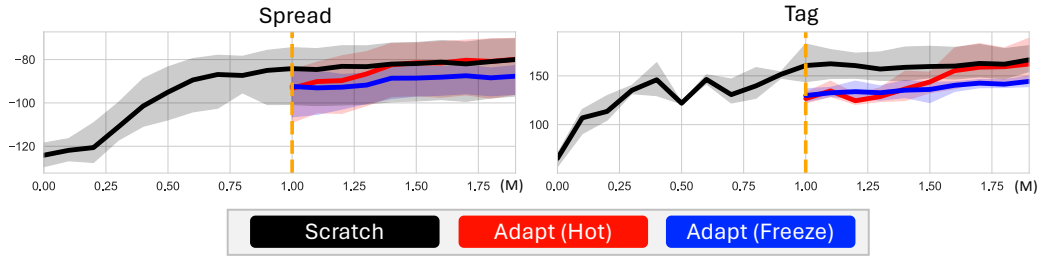


Figure 6. Episode returns for three optimizations (Y-axis) for 2M time steps (X-axis). Adaptations are started at the 1M time step. The results indicate that updating both policies (red) is better than freezing blue). The shaded regions are confidence intervals obtained by three random seeds.

As mentioned, MARL has more instability than a single RL agent. We observe more failure cases in training when freezing parameters of a base policy. Figure 7 shows the adaptation performance for seven message types after 1M time steps. When the parameters of base policies are not updated, the performance in step 2 is even less than the performance at step 1M (the last checkpoint of the base policy), indicating that the parameters of adaptation policies may have failed at optimizing given the base policy. Therefore, the best optimization setting for performance is training both networks, base policy, and adaptation policy.

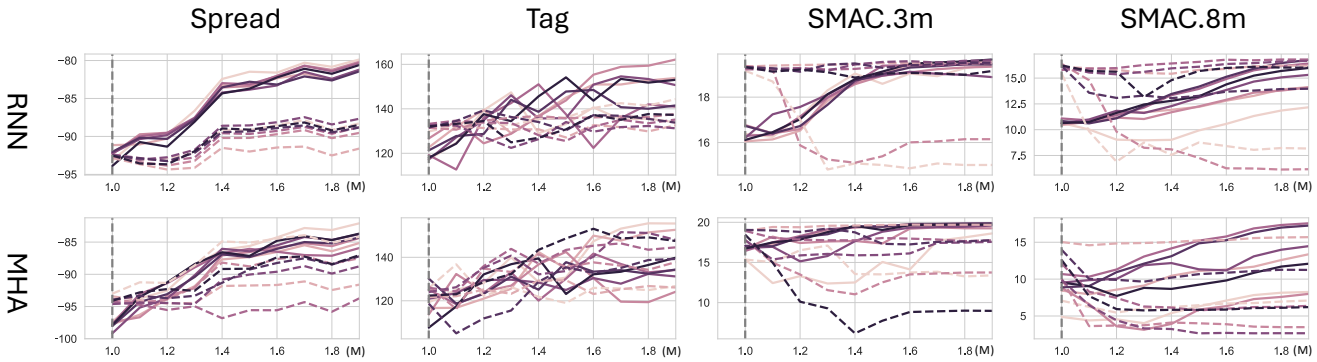


Figure 7. Episode returns for message types. The solid lines are the cases when updating base and adaptation policies, and the dashed lines are cases when the base policies are not updated. Each color represents different message types in $\mathcal{S}_{\text{Comb}}$. When freezing, the performance drops after the 1M time step, indicating that updating both networks is robust for message types.

When the base policy is updated, the communication implicitly contributes to the base policy. However, the two action branches still infer actions conditionally from observation and messages, ensuring separate inferences. In addition, this separation can reduce the training time when multiple MARL adoptions from a single RL are required. For example, when we want to train MARL agents in 10 communication settings, having a base policy will reduce the overall training time as MAAF requires only adaptation training.

7.2. Comparison to Baselines

Figure 8 shows the performance of MAAF-RNN and MAAF-MHA compared to other MARL baselines in four environments. MAAF variants perform better than other baselines for environments, spread, and tag, indicating that MAAF has competitive performance to baseline algorithms. For the comparison of message pooling methods, RNN-based pooling shows slightly better adaptation than the attention-based one. One reason is that the RNN-based pool is better than the attention-based one when the agents are fixed and small. Nonetheless, we acknowledge that the message pooling methods could be environment-specific, and training attention might benefit the message passing, as shown in TarMAC [11]. For example, MAAF-MHA performs better than RNN-based in both SMAC environments.

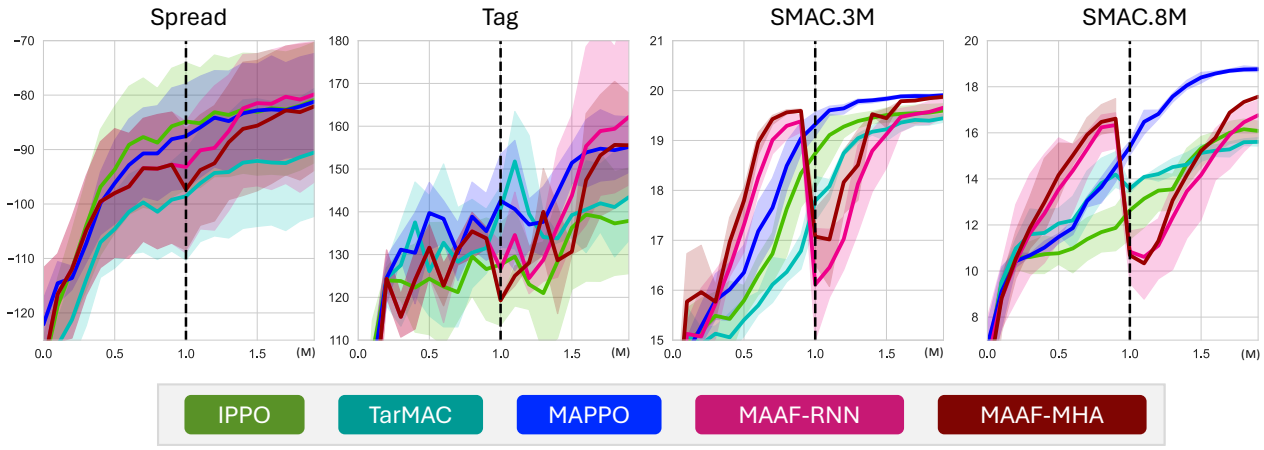


Figure 8. Episodic return while training five MARL agents for the best hyperparameter settings and three random seeds. We observe the competitive performance of MAPPO for all environments. In the tag environment, where communication between agents is crucial to tracking moving agents, the proposed MAAF variants perform better than all the baselines. However, MAAF shows weak results in SMAC environments where the number of actions is too large to be adapted. Therefore, MAAF shows a large gap to MAPPO in SMAC.8M, which has eight agents.

MAAF trains base policies until 1M and combines actions from message communication. At this time step, the output of a noise state policy is added to the base policy. Therefore, we observe a performance drop at 1M time step. In the case of SMAC 3M and 8M, we observed a significant drop in performance. This is because SMAC has more feasible candidate actions (more than 100, while MPE has less than ten actions). Therefore, although a base policy offsets the probabilistic distribution of actions, adaptation with random policy at 1M has a large influence. That is, the logits of adapted policy $\hat{\phi}$ for action a is the form of $\log \hat{\phi}(a) = \log \pi(a) + \log \phi(a)$ with noisy $\phi \sim \mathcal{N}(0, \sigma)$, and the base policy π is largely distracted as the size of action space increases. Similarly, the base policy of environments is more confident and is less affected by the randomly initialized adaptation policy. Therefore, initializing the adaptation policy ϕ with a small σ may help to adapt the adaptation policy. We leave additional optimization strategies for MAAF as future work.

We additionally evaluate MARL methods trained at the last checkpoint for 100 episodes. Table 1 shows the average evaluation returns. In addition, we provide MARL training settings on whether communication between agents exists and whether the value function uses shared observation. In tag and SMAC.3M, MAAF achieves the best evaluation results among the four models, even outperforming the value sharing (MAPPO). Compared to full communication TarMAC, MAAF-RNN achieves higher performance in all environments. Therefore, the proposed methods work when communication provides better performance when communication is essential.

Table 1. Average test return of four methods over 100 episodes. We highlight the best method in each environment with bold text. The first communication (F.C.) refers to whether messages are communicated during the first 50% of the training process, while the second communication (S.C.) indicates message passing during the last 50% of the training process. Value sharing (V.S.) refers to whether cooperative agents share observations in the value function of PPO. MAAF demonstrates an average improvement of 7.66% compared to the full communication approach used in TarMAC.

Method	F.C.	S.C.	V.S.	Spread	Tag	SMAC.3M	SMAC.8M
MAPPO	X	X	O	−66.65	116.87	18.18	**17.95**
IPPO	X	X	X	**−62.86**	106.07	18.27	15.77
Tarmac	O	O	X	−80.59	114.30	18.67	14.54
MAAF-RNN (Ours)	X	O	X	−65.37	**117.60**	**18.80**	15.73

7.3. Adaption for Different Message Types

Message channel types significantly affect cooperation performance. Although we may construct environment-specific message types, such as target locations in a predator-prey environment, manual construction of appropriate messages is non-trivial. This work considers three model-agnostic message types: observation, sampled action from the base policy, and probability distribution over actions. We concatenate messages to construct a unified-length representation for the combination of messages and reduce the dimension by 8. Figure 9 shows box plots of performances for MAAF-RNN. In the spread environment, there is little difference between different action types. This is because the environment does not depend on communication, as shown in Table 1, where IPPO, which has no communication, shows the best performance. In the case of the SMAC environment, simple message types, observations, and sampled actions show superior performance.

To conduct an ablation study on the complexity of environments, we train MAAF-RNN for tag environments with different numbers of predator agents (named by Tag-n where n is the number of agents). The number of prey agents is fixed at 1. Figure 10 shows the performance for each environment for different message types. In the Tag-2 environment, message types of single elements $\{o, a, p\}$ show superior performance than combined, such as $\{ap\}$. On the other hand, the performance in the Tag-3 environment

increases as the message types become complex. The results indicate that searching for the best message input is necessary in communication.

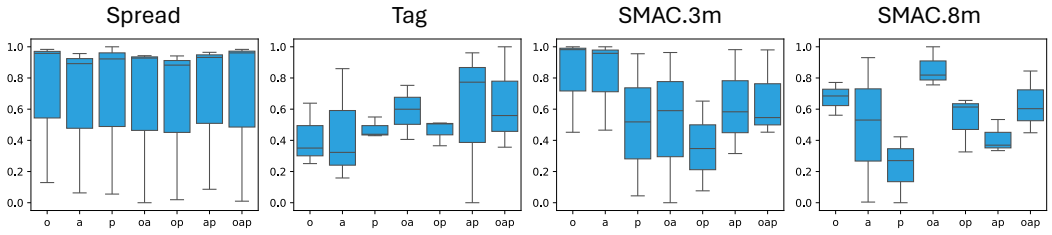


Figure 9. Box plots of normalized returns for different environments. Each plot is obtained by running three seeds. In a spread environment, we observed no difference between message types, indicating that the contribution of messages is weak. In the tag environment, the performance increases as communication message types become complex. In SMAC environments, simple messages, such as only having observation or action, provide better performance than having complex message types. These results indicate that feature selection on message types is necessary for MARL communication.

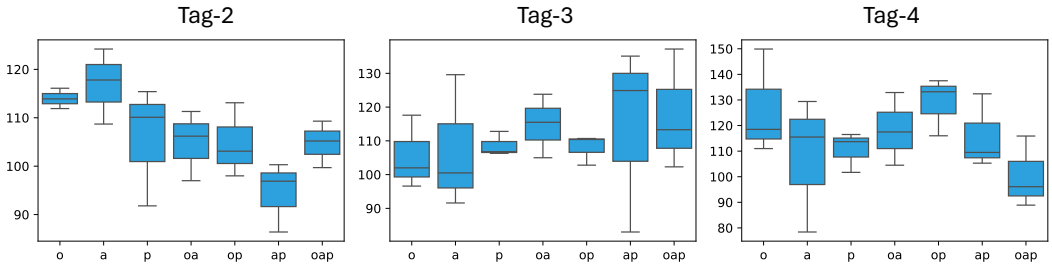


Figure 10. Box plots of returns of tag environments for different numbers of predators (the number in the naming represents the number of predators). Each plot is obtained by running three seeds. In Tag-2, the performance drops as the communication channel includes more message types. In Tag-3, the performance increases as the complexity of the communication channel increases. The action information is especially beneficial, and the best performance is obtained with both action and probability. This correlation between the complexity of message channels and environments no longer holds in Tag-4. Therefore, searching for the optimal combination might be necessary.

7.4. Joint Interpretation of Base and Adapted Policies

In this section, we propose two visualizations utilizing base and adapted policies:

1. Separate trajectories obtained by a base and adapted policies.
2. A trajectory is obtained by a policy while the other provides counterfactual action.

Figure 11 shows the trajectories of two types of actions: base and adapted. Recall that base and adaptation policies have different branches and combine the outputs only at the levels of action distribution by adding logit. The adapted policy can be interpreted as the direction of residual action. This visualization provides users insights (if visualized) by showing how communication changes the original trajectory.

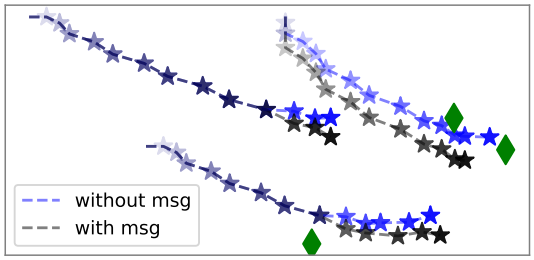


Figure 11. Trajectories for the base and adapted policies (with message cases) in the spread environment. Black and blue stars are the locations of three agents with and without communication, respectively. The darker color represents later time steps. Diamond is the target location where three agents (represented by stars) must be close to obtain rewards while separating each other.

Rather than a trajectory-level comparison, we can visualize a single-point analysis. Figure 12 shows trajectories obtained by sampling actions with a base policy and visualizing an action with a maximum likelihood for an adapted policy. This visualization provides how the counterfactual case (having messages) can change the action direction. We acknowledge that visualization is possible only when the environment is simple enough. These interpretations may not work for the cases of SMAC, where the more diverse actions are conducted.

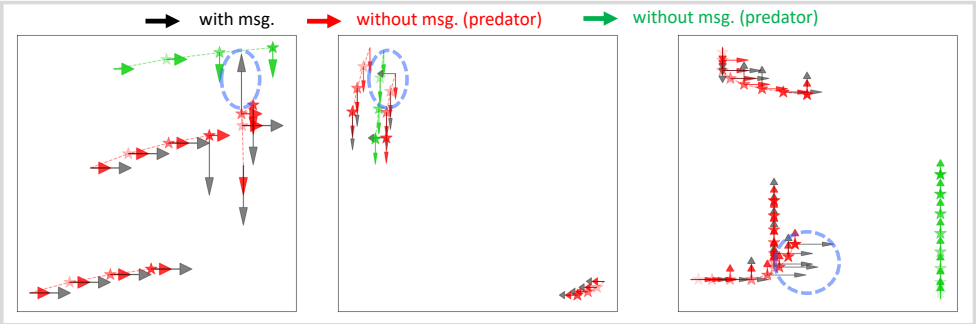


Figure 12. Counterfactual interpretation of actions following the trajectory of a base policy. Each panel shows the trajectories of an episode in the Tag environment, featuring two predators and one prey. The trajectories are represented by stars, with darker colors indicating later time steps. Green arrows represent the prey's actions, while red arrows show the actions of the predators. The gray arrows denote counterfactual actions generated through communication between predators. This setup allows us to interpret actions without communication and compare them to cooperative actions facilitated by communication.

8. Discussion

The proposed MAAF integrates message contributions linearly into action logits. MAAF outperforms baseline MARL methods in experiments, offering two key advantages: (1) At the early training stage, learning observation-dependent actions provides stability, even if performance is initially constrained to a single RL setting. (2) The communication branch facilitates further training, helping to balance the two action branches.

The performance gap observed for combinatorial message types highlights the need for adaptation. A common belief about neural networks is that they can filter out irrelevant features, making combinatorial search seemingly unnecessary. There are several reasons why message types show a considerable performance gap. One reason is aggregation methods, RNN, or self-attention (MHA), which includes more complex operations processing messages from n-agents. This complexity makes it challenging to train effective filtering in MARL. Therefore, choosing appropriate message types as an inductive bias may be more effective than using all types as input features. This work demonstrates the benefits of separating message contributions but does not explore the impact of advanced communication protocols like TarMAC or CommFormer. Since the proposed message adaptation is compatible with different communication protocols, we anticipate that these advanced protocols could yield similar performance improvements.

Separating action branches can lead to slower or less stable training when adaptation is unstable, as observed in the SMAC 8M environment. To address this, future research should focus on developing appropriate optimization strategies. Additionally, regularizing the discrepancy between the base policy and the adapted policy could help stabilize the action distribution after communication, which may be beneficial for addressing safety concerns. Another notable limitation is the assumption that message features are inherently noisy and unstable, which states that message features are noisy and unstable. This may not be valid in advanced communication architectures, such as graph-based CommFormer and sparse attention-based IMGS-MAC. Further research is needed to explore the effects of MAAF on these advanced communication methods.

9. Conclusions

We propose a novel Message Adaptation Action Framework (MAAF) that separates the communicative training process into base policy and adapted policy training. Our experiments demonstrate that this separation leads to better performance than other communication settings. Additionally, we explore the use of model-agnostic message types within MAAF, highlighting performance variations across different message types. By employing base and adapted policies, we provide interpretable methods based on trajectory analysis and counterfactual action sampling. Unlike previous MARL approaches that design communication protocols among agents, our framework focuses on the information flow of observation and message-conditioned actions.

Author Contributions: Conceptualization, B.P. and J.C.; methodology, B.P.; software, B.P.; validation, J.C.; formal analysis, B.P.; investigation, B.P. and J.C.; resources, B.P. and J.C.; data curation, B.P.; writing—original draft preparation, B.P.; writing—review and editing, B.P. and J.C.; visualization, B.P.; supervision, J.C.; project administration, J.C.; funding acquisition, J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Unmanned Swarm CPS Research Laboratory Program of Defense Acquisition Program Administration and Agency for Defense Development (UD220005VD).

Data Availability Statement: The data presented in this study are openly available in https://github.com/fxnnxc/MAAF.

Conflicts of Interest: Author Jaesik Choi was employed by the company INEEJI. The remaining author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
RL	Reinforcement Learning
MARL	Multi-Agent Reinforcement Learning
IoT	Internet of Things
UGV	Unmanned Ground Vehicle
MAAF	Message Action Adapter Framework
MDP	Markov Decision Process
DQN	Deep Q-Network
SAC	Soft Actor–Critic
A2C	Asynchronous Actor–Critic
PPO	Proximal Policy Optimization
CTDE	Centralized Training and Decentralized Execution
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
QMIX	Q-Mixing
HAPPO	Heterogeneous-Agent Proximal Policy Optimisation
PEARL	Probabilistic Embeddings for Actor-critic RL
CaDM	Context-aware Dynamics Model
TarMAC	Targeted Multi-Agent Communication
I2C	Individually Inferred Communication
IMGS-MAC	Information Maximizing Gated Sparse Multi-Agent Communication
CAMA	Complementary Attention for Multi-Agent reinforcement learning
MLP	Multi-Layer Perceptron
MDP	Markov Decision Process
SMAC	StarCraft Multi-Agent Challenge
IPPO	Independent Proximal Policy Optimization
MAPPO	Multi-Agent Proximal Policy Optimization
RNN	Recurrent Neural Network
GRU	Gated Recurrent Unit

KL	Kullback–Leibler
CaDM	Context-Aware Dynamics Model
MPE	Multi-Particle Environment

Appendix A. Implementation Details

Figure A1 shows the code implementation of MAAF action logits.

Pseudo-code for Base and Adapter Actor

```
# Base Actor computes logits based only on observation
def compute_base_action_logit(observation):
    # Step 1: Pass observation through backbone network
    base_features = backbone(observation)

    # Step 2: Compute action logits
    base_logits = base_controller(base_features)

    return base_logits

# Adapter Actor computes logits
# based on observation and message
def compute_adapter_action_logit(observation, message):
    # Step 1: Pass observation through backbone network
    base_features = backbone(observation)

    # Step 2: Pass message through adapter backbone
    adapter_features = adapter_backbone(message)

    # Step 3: Compute logits from both base and adapter
    base_logits = base_controller(base_features)
    adapter_logits = adapter_controller(adapter_features)
    final_logits = base_logits + adapter_logits

    return final_logits
```

Figure A1. Pseudo-code for computing action logits in two configurations: (1) Base Actor, which computes logits solely from observations, and (2) Adapter Actor, which incorporates both observations and messages. In the Adapter Actor computation, logits from the base and adapter networks are linearly combined to produce the final action logits.

References

1. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
2. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Dębiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with large scale deep reinforcement learning. *arXiv* **2019**, arXiv:1912.06680.
3. Rasal, S. Llm harmony: Multi-agent communication for problem solving. *arXiv* **2024**, arXiv:2401.01312.
4. Busoniu, L.; Babuska, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2008**, *38*, 156–172. [CrossRef]
5. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: New York, NY, USA, 2021; pp. 321–384.
6. Oroojlooy, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.* **2023**, *53*, 13677–13722. [CrossRef]
7. Park, B.; Kang, C.; Choi, J. Cooperative Multi-Robot Task Allocation with Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 272. [CrossRef]
8. Li, T.; Zhu, K.; Luong, N.C.; Niyato, D.; Wu, Q.; Zhang, Y.; Chen, B. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 1240–1279. [CrossRef]

9. Wang, W.Z.; Shih, A.; Xie, A.; Sadigh, D. Influencing towards stable multi-agent interactions. In Proceedings of the Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022; pp. 1132–1143.
10. Lin, J.; Dzeparoska, K.; Zhang, S.Q.; Leon-Garcia, A.; Papernot, N. On the robustness of cooperative multi-agent reinforcement learning. In Proceedings of the 2020 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 21 May 2020; IEEE: New York, NY, USA, 2020; pp. 62–68.
11. Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; Pineau, J. Tarmac: Targeted multi-agent communication. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 1538–1546.
12. Singh, A.; Jain, T.; Sukhbaatar, S. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv* **2018**, arXiv:1812.09755.
13. Sokar, G.; Agarwal, R.; Castro, P.S.; Evci, U. The dormant neuron phenomenon in deep reinforcement learning. In Proceedings of the ICML'23—40th International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023.
14. Kada, B.; Khalid, M.; Shaikh, M.S. Distributed cooperative control of autonomous multi-agent UAV systems using smooth control. *J. Syst. Eng. Electron.* **2020**, *31*, 1297–1307. [CrossRef]
15. Garcıa, J.; Fernández, F. A comprehensive survey on safe reinforcement learning. *J. Mach. Learn. Res.* **2015**, *16*, 1437–1480.
16. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
17. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
18. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
19. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
20. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390.
21. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284.
22. Zhang, K.; Yang, Z.; Liu, H.; Zhang, T.; Basar, T. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; PMLR, 2018; Dy, J., Krause, A., Eds.; Volume 80, pp. 5872–5881.
23. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.
24. Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; Mordatch, I. Emergent Tool Use From Multi-Agent Autocurricula. *arXiv* **2020**, arXiv:1909.07528.
25. Jaques, N.; Lazaridou, A.; Hughes, E.; Gulcehre, C.; Ortega, P.; Strouse, D.; Leibo, J.Z.; De Freitas, N. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 3040–3049.
26. Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; Yang, Y. Multi-agent reinforcement learning is a sequence modeling problem. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 16509–16521.
27. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, Virtual Event, Austria, 3–7 May 2021.
28. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; IEEE: New York, NY, USA, 2020; pp. 737–744.
29. Li, D.; Yang, Y.; Song, Y.Z.; Hospedales, T. Learning to generalize: Meta-learning for domain generalization. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
30. Khetarpal, K.; Riemer, M.; Rish, I.; Precup, D. Towards continual reinforcement learning: A review and perspectives. *J. Artif. Intell. Res.* **2022**, *75*, 1401–1476. [CrossRef]
31. Schweighofer, N.; Doya, K. Meta-learning in reinforcement learning. *Neural Netw.* **2003**, *16*, 5–9. [CrossRef]
32. Rakelly, K.; Zhou, A.; Finn, C.; Levine, S.; Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5331–5340.
33. Lee, K.; Seo, Y.; Lee, S.; Lee, H.; Shin, J. Context-aware dynamics model for generalization in model-based reinforcement learning. In Proceedings of the International Conference on Machine Learning, Virtual Event, 13–18 July 2020; pp. 5757–5766.
34. Guo, Y.; Wu, Q.; Lee, H. Learning action translator for meta reinforcement learning on sparse-reward tasks. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 22 February–1 March 2022; Volume 36, pp. 6792–6800.
35. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
36. Sharma, A.; Gu, S.; Levine, S.; Kumar, V.; Hausman, K. Dynamics-aware unsupervised discovery of skills. *arXiv* **2019**, arXiv:1907.01657.

37. Guan, C.; Chen, F.; Yuan, L.; Wang, C.; Yin, H.; Zhang, Z.; Yu, Y. Efficient Multi-agent Communication via Self-supervised Information Aggregation. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 1020–1033.
38. Wu, S.A.; Wang, R.E.; Evans, J.A.; Tenenbaum, J.B.; Parkes, D.C.; Kleiman-Weiner, M. Too Many Cooks: Bayesian Inference for Coordinating Multi-Agent Collaboration. *Top. Cogn. Sci.* **2021**, *13*, 414–432. [CrossRef] [PubMed]
39. Liu, R.; Yang, R.; Jia, C.; Zhang, G.; Yang, D.; Vosoughi, S. Training socially aligned language models on simulated social interactions. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
40. Ding, Z.; Huang, T.; Lu, Z. Learning individually inferred communication for multi-agent cooperation. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 22069–22079.
41. Hu, S.; Shen, L.; Zhang, Y.; Tao, D. Learning Multi-Agent Communication from Graph Modeling Perspective. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024.
42. Karten, S.; Tucker, M.; Kailas, S.; Sycara, K. Towards true lossless sparse communication in multi-agent systems. In Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA), London, UK, 29 May–2 June 2023; IEEE: New York, NY, USA, 2023; pp. 7191–7197.
43. Shao, J.; Zhang, H.; Qu, Y.; Liu, C.; He, S.; Jiang, Y.; Ji, X. Complementary attention for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Honolulu, HI, USA, 23–29 July 2023; pp. 30776–30793.
44. Knight, J.N.; Anderson, C. Stable reinforcement learning with recurrent neural networks. *J. Control Theory Appl.* **2011**, *9*, 410–420. [CrossRef]
45. Lambert, N.; Wilcox, A.; Zhang, H.; Pister, K.S.J.; Calandra, R. Learning Accurate Long-term Dynamics for Model-based Reinforcement Learning. In Proceedings of the 2021 60th IEEE Conference on Decision and Control (CDC), Austin, TX, USA, 14–17 December 2021; pp. 2880–2887. [CrossRef]
46. Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv* **2017**, arXiv:1706.06083.
47. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994, Proceedings of the Eleventh International Conference, Rutgers University, New Brunswick, NJ, USA, 10–13 July 1994*; Elsevier: New York, NY, USA, 1994; pp. 157–163.
48. Huang, S.; Dossa, R.F.J.; Ye, C.; Braga, J.; Chakraborty, D.; Mehta, K.; Araújo, J.G. CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms. *J. Mach. Learn. Res.* **2022**, *23*, 1–18.
49. Mordatch, I.; Abbeel, P. Emergence of Grounded Compositional Language in Multi-Agent Populations. *arXiv* **2017**, arXiv:1703.04908. [CrossRef]
50. Samvelyan, M.; Rashid, T.; de Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.J.; Hung, C.M.; Torr, P.H.S.; Foerster, J.; Whiteson, S. The StarCraft Multi-Agent Challenge. *arXiv* **2019**, arXiv:1902.04043.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.