Full Communication Memory Networks for Team-Level Cooperation Learning

Yutong Wang[1], Yizhuo Wang[1] and Guillaume Sartoretti[1*]

[1*]Mechanical Engineering, National University of Singapore, 9 Engineering Drive 1, Singapore, 117575, Republic of Singapore.

*Corresponding author(s). E-mail(s): mpegas@nus.edu.sg;
Contributing authors: e0576114@u.nus.edu; wy98@u.nus.edu;

Abstract

Communication in multi-agent systems is a key driver of team-level cooperation, for instance allowing individual agents to augment their knowledge about the world in partially-observable environments. In this paper, we propose two reinforcement learning-based multi-agent models, namely FCMNet and FCMTran. The two models both allow agents to simultaneously learn a differentiable communication mechanism that connects all agents as well as a common, cooperative policy conditioned upon received information. FCMNet utilizes multiple directional Long Short-Term Memory chains to sequentially transmit and encode the current observation-based messages sent by every other agent at each timestep. FCMTran further relies on the encoder of a modified transformer to simultaneously aggregate multiple self-generated messages sent by all agents at the previous timestep into a single message that is used in the current timestep. Results from evaluating our models on a challenging set of StarCraft II micromanagement tasks with shared rewards show that FCMNet and FCMTran both outperform recent communication-based methods and value decomposition methods in almost all tested StarCraft II micromanagement tasks. We further improve the performance of our models by combining them with value decomposition techniques; there, in particular, we show that FCMTran with value decomposition significantly pushes the state-of-the-art on one of the hardest benchmark tasks without any task-specific tuning. We also investigate the robustness of FCMNet under communication disturbances (i.e., binarized messages, random message loss, and random communication order) in an asymmetric collaborative pathfinding task with individual rewards, demonstrating FMCNet's potential applicability in real-world robotic tasks.

Keywords: Multi-Agent System; Reinforcement Learning; Differentiable Communications; Decentralized Cooperation

1 Introduction

Recent developments in single-agent Reinforcement Learning (RL) as well as the appearance of cutting-edge neural architectures have fueled fast progress in Multi-Agent Reinforcement Learning (MARL). MARL has achieved great success in multi-player and video games [1, 2], and showed promises for a variety of practical applications, including motion planning for autonomous vehicles [3, 4] and distributed multi-robot control [5, 6]. However, achieving decentralized and scalable cooperation among agents remains a key challenge in MARL. For instance, MARL agents often encounter partially-observable environments in real-life tasks, where learning effective cooperative policies merely from their own limited knowledge/memory can be challenging or outright impossible. Partial observability further affects the team's performance, since in many cases the intentions of other agents are often unknown or unmodeled, which prevents agents from recognizing teammates and thus often drives them to (suboptimally) treat each other as dynamic obstacles. One solution to mitigate these issues is to allow agents to communicate explicitly within the team to enhance agents' knowledge/observations, fostering true joint cooperation at the team-level. In other words, communication can allow autonomous agents to truly act as a unified group rather than as a collection of individuals.

This work contributes to the recently booming field of *Communication learning* (CL) [7, 8], in which agents are allowed to share information through learned communication mechanisms. The main goals of recent works in this area, such as SchedNet [9], G2ANet [10], ATOC [11], are to 1) learn the content of the continuous-valued messages sent/utilized by the team via gradient backpropagation similar to standard MARL, and 2) choose whom to communicate with and at which timesteps, to decrease the overall communication burden and filter out communication noise in large teams (i.e., the *chatter* problem). However, as a result, these works often do not take full advantage of information from all agents, thus limiting cooperation levels at the team level, and often lack robustness investigations, which may limit their applicability under real-life conditions, such as robotics deployment with noisy/unreliable communications, or with binary (digital) messages.

In this paper, we focus on the class of problems where global communications are available (but might be unreliable) and introduce two new communication-based models, called Full Communication Memory Net (FCMNet) and Full Communication Memory Transformer (FCMTran). These two models both enable agents to concurrently learn a global and differentiable communications mechanism as well as a common, cooperative policy conditioned upon received information. FCMNet utilizes the hidden states and

cell states of multiple parallel directional Long Short-Term Memory (LSTM) chains as messages transmitted among agents (i.e., multi-hop communications). That is, at every timestep, each agent receives and utilizes multiple messages that are generated based on the current observations of all other agents. Pushing this idea further, FCMTran uses a more advanced neural network structure, the transformer, as the communication channel. The encoder of the modified transformer in FCMTran replaces the LSTM units of FCNet to fuse messages sent by multiple agents into one message in a scalable and input-order-independent manner. Different from FCMNet, the message content in FCMTran is self-generated by the transformer, starting from a zero-vector input at the first step of each task. That is, in FCMTran, the output of the transformer at the previous step is used as the only input of the transformer at the current step, thus allowing agents to obtain information across timesteps. The resulting, realistic time delay between sending and receiving messages also relaxes the assumptions towards real-life deployments. By relying on our proposed neural structures and sharing weights among agents in a discrete manner throughout our framework, the differentiable communication channels of FCMNet and FCMTran can be trained using the losses from all agents, thus exhibiting more efficient high-cooperation policies than existing CL methods. We experimentally evaluate our proposed models on a range of unit micromanagement tasks in the StarCraft II Multi-Agent Challenge with shared team rewards [12]. There, our evaluation results show that FCMNet and FCMTran outperform recently published CL methods and value decomposition methods in almost all tested tasks. After combining our models with value decomposition techniques from QMIX [13] and VDN [14], their performances is further improved. In particular, we show that FCMTran with value decomposition significantly outperforms the state-of-the-art (SOTA) in one of the hardest benchmark tasks without any task-specific tuning/tricks. Finally, we investigate the robustness of FCMNet under realistic communication interference (i.e.,binary messages, random message loss, and randomized communication orders) on a partially-observable multi-agent pathfinding task with individual rewards [15]. Our findings demonstrate that FCMNet exhibits a type of natural resilience to such communication interference, suggesting promises for deployments in tasks with real-life communication constraints.

2 Related Work

2.1 Communication Learning

CL is one of the subareas of MARL, in which agents learn to share information within the team, thus effectively supplementing the knowledge available to individual agents (e.g., from local sensing) and mitigating the hazards of partial observability. Foerster et al. [16] first proposed two types of learnable communication mechanisms, namely RIAL and DIAL. While RIAL treats communication as an additional action to be chosen, DIAL learns real-valued

messages passed between agents during centralised learning by backpropagating the gradients from the recipient agent(s) through the differentiable communication channel. Because of this richer feedback, DIAL was shown to substantially outperform RIAL on complex tasks. Following a similar direction to DIAL, CommNet [17] proposed to use the average hidden state of all other agents as the message to be transmitted, and allows several rounds of agent communication at each timestep. BiCNet [18] relied on bidirectional RNNs to build communication channels among agents, thus sequentially integrating information from all agents in an equal structure. More recently, Kong et al. [19] constructed a RNN-based hierarchy communication, in which each low-level agent reasons on a local yet focused scale and a single high-level agent reasons at the global level.

All of the aforementioned methods necessitate constant communication between agents, with the communication target either being predefined or broadcast to all agents. Learnable dynamic communication architectures have recently attracted the community's interest due to the fact that such architectures selectively reduce the amount of information that needs to be shared, thereby reducing the communication overhead and limiting the number of (potentially contradictory) messages that can be shared as team sizes increase. For example, G2ANet [10], MAGIC [20] and ATOC [11] all use an attention mechanism to indicate whether communication is required and how to combine incoming information towards cooperative decision making. SchedNet [9] enables agents to determine which agents' encoded messages are valuable to broadcast within the team at each timestep by learning the importance of each agent's partially observed information. However, such selective dynamic communication architectures inevitably perform poorly in problems that may require (or can provide) global communication between agents, such as the tasks examined in this work, because the available information to the team naturally decreases with selective communications.

2.2 Transformers in Reinforcement Learning

In recent years, the transformer architecture [21] has revolutionized the learning paradigm across numerous fields of artificial intelligence and proven superior performance over the convolutional neural network and RNN. Inspired by the success of transformers in other domains (mainly natural language processing), there has been a surge of interest in applying these scalable, attention-based models to RL [22].

GTrXL [23] first proposed to modify the Transformer-XL architecture to substantially improve the stability and learning speed of the original transformer variant in online RL. The Transformer-XL structure proposed in GTrXL is employed as a memory architecture to process trajectory data. In order to comprehend the global context of the robot's belief, graph-based transformer architectures have also been proposed for informative/exploratory path planning tasks [24, 25]. In addition, recent works motivated by offline RL have demonstrated that the transformer architecture can directly serve as a model

for sequential decision-making based on fixed large-scale offline data. Chen et al. [26] first showed that, by conditioning on expected returns, past states, and actions, transformer-based models can effectively generate future desired trajectories without temporal difference learning or dynamic programming in online RL. Following this work, for solving visual input tasks, StARformer [27] extracts state-action-reward representations within a short temporal window and uses these representations to execute self-attention across the entire sequence. The application of transformers in MARL has not been fully explored to date, compared to computer vision and natural language processing. But since transformers have demonstrated significant performance gains compared to RNN in other domains, we hypothesize that transformer-based communication mechanisms should also outperform previous RNN-based communication mechanisms [18, 19].

3 Background

3.1 Proximal Policy Optimization Algorithms

Proximal Policy Optimization (PPO) [28] is a well-known on-policy RL algorithm with an actor-critic structure and trained by policy gradient. Different from standard policy gradient methods that usually perform one gradient update per data sample, PPO adds a clipped probability ratio to the optimisation objective of the actor, which helps prevent destructively huge policy updates, enables multiple epochs of minibatch updates, hence dramatically increasing sample efficiency. Specifically, PPO adjusts the weights θ of the actor network to maximize the loss Eq.(1), resulting in monotonic improvements in policy performance:

$$L^{CLIP}(\theta) = \widehat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right], \tag{1}$$

where $r_t(\theta)$ denotes the clipped probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$, with π_θ and $\pi_{\theta_{old}}$ the new and old policies of the agent, respectively, ϵ a clipping hyperparameter (often set to $\epsilon = 0.2$), and $\hat{A}_t$ the truncated version of the generalized advantage function. By taking the minimum of the clipped and unclipped objectives, PPO effectively bounds the rate of change of the agent's policy. In addition, PPO also adds a policy entropy term to the final actor loss, which encourages exploration by preventing premature convergence to inferior, deterministic policies.

In this paper, while both FCMNet and FCMTran are compatible with other training algorithms, we rely on the multi-agent version of PPO to train agents in a centralized training and decentralized execution (CTDE) framework.

3.2 Transformer

The structure of the Transformer [21] model consists of two main parts: the encoder and the decoder. The encoder is responsible for encoding the input

sequence into a set of hidden representations, while the decoder generates the target sequence using the encoder's output and previous outputs. Both the encoder and decoder are usually constructed by stacking multiple identical blocks. Each encoder block has two main sub-layers: a multi-head self-attention module and a position-wise fully connected feed-forward network. To enhance the learning capacity and training stability of this deep network, a residual connection [29] is often applied around each sub-layer, followed by layer normalization [30]. Compared to the encoder block, each decoder block consists of three main sub-layers. There, alongside the multi-head self-attention module and the position-wise fully connected feed-forward network, a cross-attention module is inserted between the two sub-layers, which performs attention on both the output of the encoder and its own previous output. Furthermore, to inject the positional information of the tokens in the sequence, a position encoding is added to the input embedding at the bottom of the encoder and decoder stacks.

3.3 Value Decomposition

In multi-agent cooperative tasks where only a single joint reward is provided to the agents, both independent Q-learning and fully centralized learning methods perform poorly due to their inability to correlate global rewards with the correct individual agent action in practice. Value decomposition methods tackle this credit assignment issue by factorizing the global, team value function into agent-specific value functions.

Vanilla Value Decomposition Networks (VDN) [14] assume the joint value function can be represented as a sum of individual value functions. Taking advantage of the state information during training, QMIX [13] further relaxes this summation assumption to a more general monotonic constraint by learning a mixing network. It adopts separate hypernetworks to produce weights and bias of the mixing network, which guarantees the monotonicity of individual value functions while conditioning joint action-value on extra state information. Based on these methods, Su et al. [31] proposed value-decomposition actor-critic (VDAC) to decompose global state-values instead of action-values, to incorporate VDN and QMIX into actor-critic algorithms. As a policy-based method, PPO also fits in the actor-critic framework, thus allowing us to combine it with VDN and QMIX to improve the cooperation learning.

4 Full Communication Memory Networks

In this section, we describe the proposed models in details. We first describe FCMNet, which is built on multiple directional LSTM chains, and where sent messages are encoded from each agent's current observation and received messages so far at that timestep. We then introduce FCMTran, which is based on the encoder of a modified transformer, and where the content of messages is self-generated by the transformer based on information from the previous timestep (starting from a zero vector input at the first step of each task).

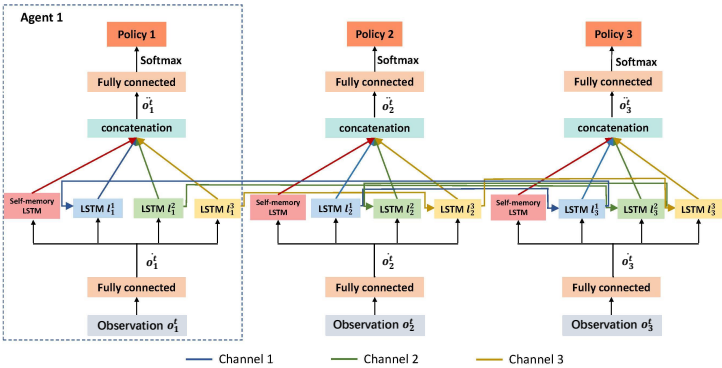


Fig. 1: Structure of the policy network in FCMNet, for an example task with three agents and therefore three parallel communication channels (arrows between agents in the middle of the network). Communications flow in the direction of these arrows, connecting the agent's LSTM units to form three parallel, one-directional communication channels. There, the output hidden state and cell state of each LSTM unit serve as messages. These messages are sequentially transmitted between agents along each channel; all communications occur in the same timestep (i.e., multi-hop communications). Most of the components of the independent critic network in FCMNet are the same as described in this figure, but the output of each agent is a single state values estimate. We use parameter sharing among agents for both the actor and critic networks, but no parameter sharing among these two networks.

We note that the two models are applicable to general multi-agent problems if the following conditions are met:

- All agents in the environment are able to communicate with each other, and communications are not hindered by any obstacles.
- For FCMNet, agents are able to have multiple rounds of communication within one timestep (multi-hop capabilities).
- For FCMTran, the time delay from sending a message to receiving/using it should be smaller than or equal to one timestep.

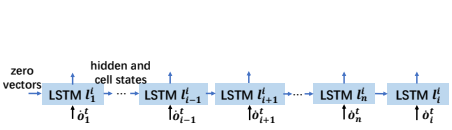
4.1 FCMNet

FCMNet is built on an actor-critic structure under the CTDE framework where both the actor and critic of each agent only transmit information in the communication layer. The remaining layers are separate, and all their parameters are shared among all team members. The CTDE framework enables the critic to obtain additional information during training, resulting in a faster and higher-quality convergence. Weight sharing also helps speed up the learning process by relying on the bulk of experience from all agents. Finally, weight sharing among agents also leads to a form of invariance in the agents' policy.

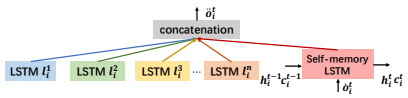
That is, FCMNet can avoid learning inefficient policies where only one agent is active and the other agents do not contribute to the team.

The neural network structure of the policy network in FCMNet is shown in Figure 1. For agent i at timestep t, its local observation o_i^t is first encoded into $\dot{o}_i^t$ via one fully connected layer, then be fed to a global multi-hop communication mechanism as the content base of the generated messages. The global multi-hop communication mechanism is the key of FCMNet. It is based on parallel directed LSTM chains that connect agents along different sequences, thus forming multiple different parallel communication channels. For a task with n agents, we assign n LSTM units to each agent and connect these LSTM units in a non-repetitive sequence to build n communication channels of length n, where each agent only has a single LSTM unit in each communication channel. Specifically, the LSTM units set of agent $i \in \{1, 2, 3 \dots n\}$ is denoted as $L_i = \left(l_i^1, l_i^2, l_i^3 \dots l_i^n\right)$. Communication channel $j \in \{1, 2, 3 \dots n\}$ is composed of a set of LSTM units,i.e., $C_j = (l_1^j, l_2^j, l_3^j \dots l_n^j)$, where l_i^j represents the LSTM unit of agent i in communication channel j. The parameters of the LSTM units in different communication channels are different, but the parameters of the LSTM units in the same communication channel are shared by all agents, i.e., the parameter of the LSTM units in L_i is different and the parameter of the LSTM units in C_j are the same. The LSTM unit can be viewed as an information extractor/encoder in FCMNet, the hidden state and cell state output by each LSTM unit are the messages communicated between agents. The hidden state and cell state of the LSTM unit of the preceding agent will be used as the initial hidden state and cell state of the LSTM unit of the subsequent agent to offer an efficient way to encode high-level information in latent space into a fixed-length message that is sequentially transmitted among agents. That is, FCMNet enables agents to learn to sequentially integrate their observations on top of information from other upstream agents in that communication channel, allowing each to receive and contribute to the ongoing information flow of the entire team.

For each agent to receive extracted information from all other agents, we introduce a simple fixed connection topology for the LSTM units in these communication channels. As shown in Figure 2a, in communication channel i, the LSTM unit of agent i is always in the last position of the channel, and other agents are connected in front of it in a fixed and non-repetitive sequence (in practice, we let channel ith follow the sequence $1, \dots, i-1, i+1, \dots, n, \mathbf{i}$) with zero vectors as initial hidden and cell states. Consequently, the message received by agent i on the ith communication channel contains information processed by all prior agents. We experimentally found that the sequence of other agents in front of the last position does not have a significant effect on the final performance of FCMNet. The final output of the communication mechanism of agent i is the concatenation of the output of its LSTM unit set $L_i = (l_i^1, l_i^2, l_i^3 \dots l_i^n)$, since the messages received by the agent in all communication channels all contain useful information, even though some of these messages contain information extracted by only a subset of the team.



(a) Communication channel i



(b) Concatenating the communication mechanism and the self-memory unit

Fig. 2: Details of the FCMNet middle layer. h_i^t and c_i^t represent the output hidden state and cell state of agent i's self-memory unit at timestep t, respectively.

As demonstrated in Figure 2b, the output of the communication mechanism is then concatenated with the output of an additional LSTM cell (self-memory unit) into $\ddot{o}_i^t$. The input hidden and cell states of the self-memory unit are its output hidden state and cell state at the previous timestep, and its input is the agent's encoded current observation $\dot{o}_i^t$. Through the self-memory unit, agents are able to integrate past information across time. Finally, $\ddot{o}_i^t$ after a fully connected layer (and a softmax layer in the policy network) is used to generate agents' policy or predicted state value at the current timestep.

Since communications occur prior to action selection, agents' decisions are indeed conditioned on their own observation as well as on messages exchanged team-wide. In addition, since transmitted messages are continuous-value vectors and communication channels are differentiable, the whole communication mechanism is trained by gradients from all agents' policy and critic losses. Such differentiable CL enables FCMNet agents to directly inform each other of exactly how received messages drove them towards better/worse actions, hence enhancing each other's action selection through backpropagation during centralized training. As a result, communication channels are trained to capture more meaningful information that reduce agents' losses, resulting in enhanced-cooperation policies. The key capabilities to identify and share relevant observation information and propagate gradients among all agents are most likely reasons why we are able to report improved performance in highly cooperative tasks, which can only be achieved when agents consensually act as a single coherent unit. Futhermore, since the final output of the communication mechanism is derived from multiple communication channels, received messages have been processed multiple times by other agents. Therefore, even if one received message contains inaccurate/noisy information, this message will likely be diluted in the set of other, more accurate messages, often still allowing efficient decision-making. In conclusion, the global, multi-hop communication features of FCMNet endow agents with a natural resistance to interference by introducing a form of messaging redundancy over the multiple parallel communication channels.

4.2 FCMTran

FCMTran also aims to augment additional knowledge to the agents through a differentiable communication mechanism trained by the losses of all agents,

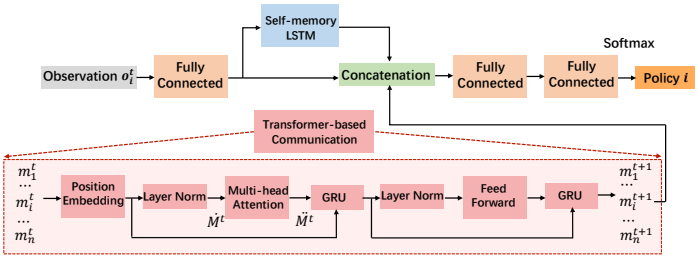


Fig. 3: Structure of the policy network for agent i at timestep t in FCMTran. The red modules belong to the transformer-based communication mechanism. The details of communication mechanism are shown in the red dashed box at the bottom of the figure. Its input is the output of the transformer encoder at the previous timestep and, its output is sent to every agent (one message per agent), to be concatenated with the output of the self-memory LSTM and the output of the previous fully connected layer.

thus mitigating limitations that arise from partial observability and enhancing team-level cooperation. Part of the neural network structure and training techniques of FCMTran are similar to that of FCMNet and enjoy the same benefits. For example, FCMTran is also based on the actor-critic structure under the CTDE framework and trained using the multi-agent version of PPO. FCMTran agents transmit information only in the communication layer of the policy net (the critic network is simplified to four fully connected layers with one self-memory unit), and other layers remain separate with weight sharing among agents. The final communication output is concatenated with the input of the communication layer and the output of the self-memory unit as the basis for decision making.

The two important differences between FCMTran and FCMNet are that: first, FCMTran uses the encoder of a modified transformer as the communication layer, in which all weights are shared among agents (instead of multiple LSTM chains with communication-channel-specific weights in FCMNet); Second, the input of the communication layer of FCMTran (i.e., the content of messages) is the output of the communication layer from the previous timestep (the first step input of each task is a zero vector) rather than the encoded current observation in FCMNet. Therefore, communication in FCMTran helps integrate history knowledge among all agents, which can be seen as an enhanced version of the self-memory LSTM unit in FCMNet, which also uses the output of the previous step as the input for the current step. Since transformers have shown considerably stronger performance over LSTM in other areas, we expect that the summary memory and message aggregation function of the transformer should also be stronger than LSTM in MARL. However, utilizing transformers in online RL also poses several unique challenges. Different from supervised learning, online RL is plagued by inefficient data utilization and limited data quantity. On the other hand, a Transformer-based model typically requires a larger amount of data for effective and

stable training. Moreover, transformer-based architectures suffer from enormous memory footprints and high computational costs, which make training and inference expensive throughout the RL learning process. Therefore, in this work, we use only one encoder block and follow the instructions of GTrXL [23] to remove dropout layers, rearrange the placement of layer normalization in the submodule, and additionally add a Gate Recurrent Unit (GRU) in place of the residual connection of the vanilla transformer to provide a "skip" path from the temporal input to the output of the transformer, thus stabilizing the training.

The neural network structure of FCMTran for agent i at timestep t is shown in Figure 3. The multi-head self-attention is the core component of the communication layer in FCMTran. The multi-head self-attention mechanism at timestep t reads:

$$\ddot{M}^t = \text{concat}\left(\text{ head }_1, \cdots, \text{ head }_h\right) W_o \tag{2}$$

$$\text{where head }_i = \text{softmax}\left(\frac{\boldsymbol{W}_q^i \dot{M}^t \cdot \left(\boldsymbol{W}_k^i \dot{M}^t\right)^\top}{\sqrt{d_K}}\right) W_v^i \dot{M}^t, \tag{3}$$

$\dot{M}^t = [\dot{m}_1^t, \ldots \dot{m}_n^t]$ is the packed matrix of all agents' messages after processing by the previous layers; W_q^i, W_k^i, W_v^i and W_o are the learnable weights used to linearly project matrices into the query, key, and value of head i, and the final attention output; d_k is the dimension of the key used as a scaling factor used to mitigate the gradient vanishing problem of the Softmax function; $\ddot{M}^t$ is the final joint output matrix from different heads (i.e., representation subspace) whose sub-vectors are then processed by the subsequent fully connected layers to be the n fused message received by the n agents (one message per agent).

Unlike other CL methods that use fixed aggregation functions (e.g., average function [17], maximum function [9]), in FCMTran, agents are able to combine multiple messages into a single message using learnable weights in an attention mechanism. Thus, complex pairwise relationships between messages can be captured, useless/harmful messages can be eliminated by giving an attention weight close to zero, and conversely, important messages can be specifically attended to. FCMNet also allows messages to be merged by learnable weights, but FCMTran is more intuitive. Furthermore, because FCMTran uses positional embedding to insert the agents' ID into the input messages, the order of communication is no longer a limitation. FCMTran allows a one timestep delay between sending and receiving messages, further relaxing the assumptions of FCMNet towards real-life deployments. Finally, FCMNet requires different weights for different communication channels, and the number of channels increases linearly with the number of agents. This feature limits the scalability of FCMNet and, when the number of agents is too large, the model size becomes excessively large as well, making optimization and complete training difficult. In contrast, all weights of FCMTran are shared among

all agents, and the size of the model is independent of the number of agents, thus solving the team size limitations and scalability problems.

5 Experiments

In this section, we first benchmark FCMNet, FCMTran, and their variants with value decomposition technique[1] against a set of value decomposition-based and communication-based baseline algorithms in a standardized partially-observable StarCraft II micromanagement environment with shared reward, called SMAC [12]. We then investigate the robustness of FCMNet under three realistic communication disturbances in a collaborative multi-agent pathfinding task with individual rewards. The experimental setup and the hyperparameters of our models are detailed in Appendix 6.

5.1 Performance Experiments

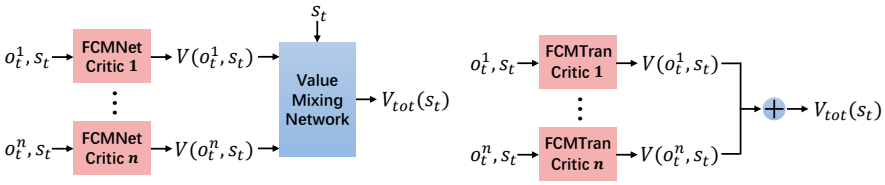
5.1.1 StarCraft II Micromanagement with Shared Reward

SMAC is a partially-observable environment with shared rewards and discrete action space built on the strategy game StarCraft II. The environment only simulates a battle between two platoons of units to assess how well independent agents are able to collaborate to solve complex skirmish tasks. In each scenario, one army is controlled by a RL algorithm, in which each unit is an independent learning agent. The opposing army is controlled by the built-in, non-learned, heuristics game AI. SMAC has been widely used in MARL research as a standard environment. Some recent work has improved algorithm performance significantly by modifying the output of the environment or proposing clever task-specific tricks [32, 33]. In order to compare our approach fairly, in this work we preserved the default setting of SMAC and did not introduce any additional tricks such as reward-shaping, death agent masking, etc. We consider the following standard SMAC tasks in our experiments, in ascending order of difficulty: $2m\_vs\_1z$, $3m$, $3s\_vs\_3z$, $3s\_vs\_4z$, $10m\_vs\_11m$, $5m\_vs\_6m$. These are all fully cooperative and homogeneous multi-agent tasks, but the units assigned to the two platoons differ. The overall objective is to eliminate enemy units, namely, maximize the win rate (the percentage of episodes in which agents defeat all enemy units within the time limit).

5.1.2 Result and Analysis

We compare FCMNet, FCMTran, FCMNet-QMIX, and FCMTran-VDN with 5 standard baselines in the field, namely CommNet [17], G2ANet [10], SchedNet [9], VDN [34] and QMIX [13] on the 6 different SMAC tasks considered. CommNet, G2ANet, and SchedNet are all differentiable CL approaches, closer to our methods. CommNet uses the average value of hidden states from all agent modules as a communication message and allows multiple rounds of

[1] The full code will be made available publicly upon paper acceptance.



(a) FCMNet with QMIX (b) FCMTran with VDN

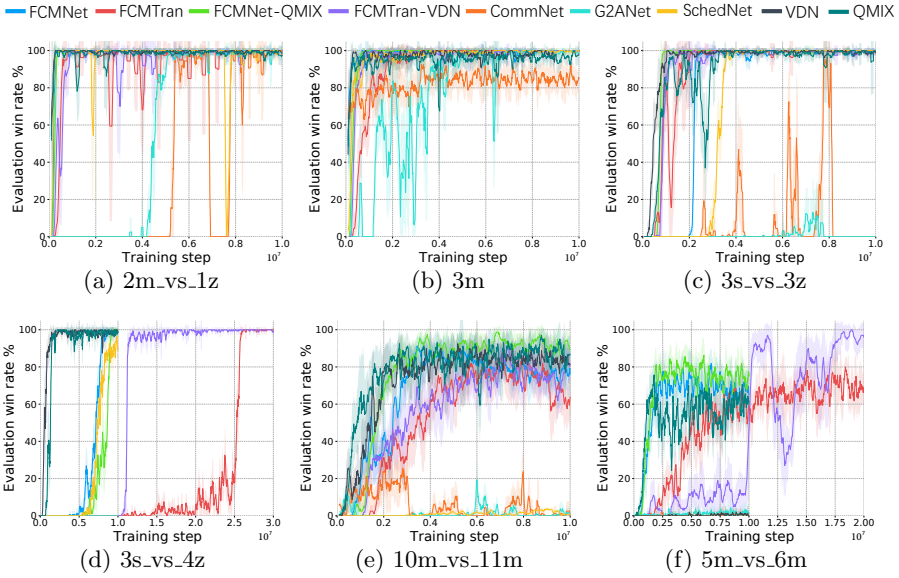
Fig. 4: Combination of value decomposition methods (QMIX and VDN) with FCMNet and FCMTran.

Table 1: Mean evaluation win rate and standard deviation on all the SMAC tasks considered after convergence of different algorithms. Results are calculated by running 16 tasks 10 times each. The score of the best-performing algorithm(s) for each task is highlighted in bold.

	2m_vs_1z	*3m*	*3s_vs_3z*	*3s_vs_4z*	*10m_vs_11m*	*5m_vs_6m*
FCMNet	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	88.1(9.9)	75.0(12.5)
FCMTran	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	79.4(7.9)	72.5(8.9)
FCMNet-QMIX	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**95.6 (4.0)**	85.6 (10.5)
FCMTran-VDN	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	90.6(7.5)	**98.9(3.8)**
CommNet	93.8(10.8)	90.6(3.1)	0.0(0.0)	0.0(0.0)	0.0(0.0)	0.0(0.0)
G2ANet	**100.0(0.0)**	**100.0(0.0)**	0.0(0.0)	0.0(0.0)	1.6(2.7)	0.0(0.0)
SchedNet	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	89.1(5.2)	0.7(0.3)	0.0(0.0)
VDN	96.9(5.4)	**100.0(0.0)**	**100.0(0.0)**	98.4(2.7)	78.1(7.0)	18.8(0.0)
QMIX	96.9(3.1)	**100.0(0.0)**	98.4(2.7)	98.4(2.7)	85.9(5.2)	59.4(3.1)

communication between agents per timestep. G2AN employs hard- and soft-attention to first determine if there is communication between two agents and then calculate the importance of the communication. SchedNet chooses k out of n agents to broadcast their encoded messages in each timestep by estimating the significance of each agent's partial observation to the team. In our experiments, we always set the hyperparameter $k = n - 1$, i.e., only one agent is unable to broadcast its messages in each timestep. We let agents in FCMNet, FCMTran, Commnet, G2ANet, and SchedNet communicate n, 1, 3, 1, and 1 time(s) per timesteps, respectively. Both VDN and QMIX belong to the class of Q-learning-based value decomposition approaches. They implement different monotonicity constraints to explicitly solve the credit assignment problem in team reward tasks but do not endow agents with communication abilities. Recent work has shown that these approaches can be combined with actor-critic algorithms to yield high-performance solutions [31]. For more details about them, we refer the reader to Section 3.3. In this work, to further improve the performance of our CL approaches, we combine our models with VDN and QMIX. Here, we only report the results of FCMNet-QMIX and FCMTran-VDN, as they show the best performance of all four possible combinations. Figure 4 briefly illustrates how VDN and QMIX are combined with the critic(s) of FCMNet and FCMTran.

The specific results are present in Table 1 and Figure 5. Overall, FCMNet, FCMTran, FCMNet-QMIX, and FCMTran-VDN all solve the four easy tasks with a 100% win rate and performe excellently in the two harder tasks.



(a) 2m_vs_1z (b) 3m (c) 3s_vs_3z

(d) 3s_vs_4z (e) 10m_vs_11m (f) 5m_vs_6m

Fig. 5: Training curves of different algorithms on SMAC tasks, showing the average win rate. The confidence interval (shaded area) shows one standard deviation over 160 evaluation episodes. We adopt the following evaluation procedure until the algorithm converges: the training is paused after every data collection during which 16 evaluation episodes are run with agents performing greedy action selection in a decentralized fashion.

Compared with FCMTran and its variants, FCMNet and FCMNet-QMIX converge faster and have better final performance when solving tasks with large team size, e.g., FCMNet has a 8.7% higher win rate than FCMTran in the $10m\_vs\_11m$ task. Furthermore, we note that the value decomposition techniques improve the performance of both FCMNet and FCMTran by nearly 10% win rate. It is worth noting that FCMTran-VDN obtained a win rate of 98.9% in the most challenging $5m\_vs\_6m$ task, which not only significantly outperformed the methods evaluated in the paper but, to the authors' limited knowledge, achieved the SOTA performance of the MARL community in this very difficult task. Moreover, all our methods outperformed the communication-based baseline by a considerable margin in all tasks. The gap between them is more pronounced for harder tasks, such as the $10m\_vs\_11m$ and $5m\_vs\_6m$ tasks, where our approaches still perform quite efficiently, while the final win rate of these communication-based baselines drops to about 0%. We believe that this is due to the fact that our models allow agents to learn how to fuse messages, which is more flexible and powerful than CommNet with fixed average operations. Additionally, in contrast to G2AN and SchedNet, we allow all agents to communicate with each other at every timestep, rather than only a selected subset of agents; such global communications seem crucial to achieve the type of near-joint decision making required in the SMAC

tasks. The two value decomposition methods performed relatively effectively in all tasks, with QMIX having a 6.5% higher win rate than FCMTran in the $10m\_vs\_11m$ task. The reason is that value decomposition provides agents with more accurate learning signals that effectively identify single-agent contributions to the team reward, which often leads to agents' easier identification of individual strategies that help improve teamwork. We also note that, although the final evaluation results of FCMTran are impressive, its training curves are more unstable. The number of timesteps required for its convergence is approximately 5 times higher than other evaluated baselines in the $5m\_vs\_6m$ task and the final convergence values are much more sensitive to random weight initialization. This suggests that further improvement techniques may still be needed to fully exploit the power of transformers in data-limited online RL.

5.1.3 Ablation Study

To fairly demonstrate the effectiveness of our two communication structures compared with a neural structure directly using the observations from all agents as input, we developed one such ablation baseline, termed Full-obs. The network structure, training algorithm, and hyperparameters of Full-obs are identical to the original FCMNet and FCMTran, except that its entire communication mechanism is completely removed, and that each agent directly uses the concatenated/joint observation of all agents as input. The evaluation results of this ablation study are shown in table 2. There, we observe that both FCMNet and FCMTran achieve a higher success rate on most tasks compared with Full-obs, supporting our claim that our two communication methods are able to capture more meaningful information, leading to enhanced team-level cooperation.

Table 2: Results of our ablation study. Mean evaluation win rate and standard deviation on SMAC tasks, after convergence of the different algorithms. Results are calculated by running 16 tasks, 10 times each. The score of the best-performing algorithm(s) for each task is highlighted in bold.

	$2m\_vs\_1z$	$3m$	$3s\_vs\_3z$	$10m\_vs\_11m$	$5m\_vs\_6m$
FCMNet	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	**88.1(9.9)**	**75.0(12.5)**
FCMTran	**100.0(0.0)**	**100.0(0.0)**	**100.0(0.0)**	79.4(7.9)	72.5(8.9)
Full-obs	**100.0(0.0)**	99.4(1.9)	**100.0(0.0)**	80.6(6.5)	65.0 (6.4)

5.2 Robustness Experiments

5.2.1 Multi-Agent Pathfinding with Individual Rewards

To further investigate the robustness of FCMNet, we consider the simple, partially-observable, cooperative multi-agent pathfinding task with individual rewards and discrete action space introduced in [15] (*Hidden-Goal Path-Finding*). There are 5 agents in this task, and each agent has a unique target location it needs to reach as soon as possible, and the target location may change randomly at every timestep. Each agent's observation includes the

location of other agents' targets but **not its own target**. Therefore, effective communication is required to solve this problem.

We also investigated the performance of FCMTran in this task using the default hyperparameters of FCMNet, however, even under normal conditions without interference, FCMTran was not able to train until convergence (i.e., never learned to solve the task). We believe this outcome may be due to the fact that transformers are inherently more difficult to optimize, and can exhibit unstable training processes; further hyperparameter tuning may be necessary, and is left to future work.

5.2.2 Binarized Messages

The communication mechanism in FCMNet is differentiable and optimized by backpropagation, where gradients flow between agents via the communication channels. This provides richer feedback to agents, so our model tends to converge faster and to higher-quality policies than reinforced CL techniques [16] that treat communication as actions, and observe their effect from subsequent rewards. However, the majority of modern communication technologies rely on discrete communication channels, for which continuous communication cannot be directly applied (i.e., bitwise/digital communications). To further examine the practicality of FCMNet under such realistic constraints, we convert the original continuous real-valued message of FCMNet (i.e., the hidden state and cell state of each agent's LSTM units) with a length of 128, into a binary message with a length of 20 by introducing a stacked autoencoder and a binarization step into the actor's communication channels.

The neural network structure between two LSTM units of FCMNet with binarized messages is depicted in Figure 6, where the weights of the autoencoder are shared among agents in the same communication channel, but can be different among different communication channels. The two-step binarization process we employ is inspired by [35–37]. In the first step, the encoder generates the desired length of outputs in the continuous interval $[-1, 1]$, using a fully-connected layer with a tanh activation function. In the second step, the binarization process produces discrete data in the set $\{-1, 1\}$ from the continuous output of the encoder:

$$b(x) = x + \epsilon \in \{-1, 1\},$$

where x represents each component of the original continuous vector, $\epsilon \in \{1-x, -x-1\}$ is a random variable distributed according to $P(\epsilon = 1-x) = \frac{1+x}{2}$ and $P(\epsilon = -x-1) = \frac{1-x}{2}$. Therefore, the complete binarization process for each scalar is:

$$B(x) = b\left(\tanh\left(\omega x + b\right)\right),$$

where ω and b are the weight and bias of the fully-connected layer with tanh activation function.

The process of message transmission is shown in Figure 6, the message sender first converts a real-valued message into a binary message through the

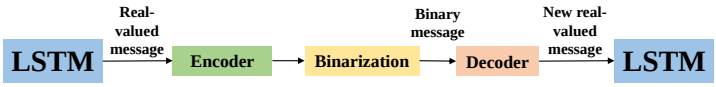
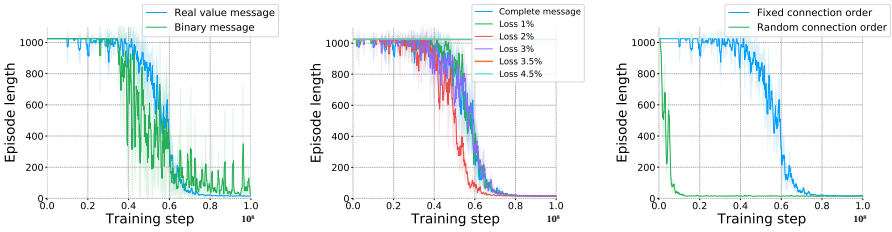


Fig. 6: Binarization/De-binarization process, implemented between each two consecutive LSTMs (i.e., between communicating agents) in the FCMNet variant with binary messages.



(a) Binarized messages (b) Lost messages (c) Random order

Fig. 7: Training curves for our robustness investigation. Evaluations are performed after every data collection, and the number of episodes per evaluation is 16. The confidence interval (shaded region) displays one standard deviation over 80 evaluation episodes. These plots demonstrate the average number of steps required to complete a pathfinding task, where shorter episodes are better. Our results indicate that FCMNet still converges to the same level of performance, even under three communication disturbances: binarized messages, (low rate of) random message loss, and randomized communication orders at each timestep.

encoding and binarization processes. The receiver agent then inputs the binary message into a symmetrical decoder to generate a new real-valued message. In accordance with the standard FCMNet architecture, the new real value message is utilized as the input hidden state and cell state of the subsequent LSTM unit.

Figure 7a shows the learning speed and final episode length of FCMNet with binarized messages in the simple multi-agent pathfinding task, which remain similar to FCMNet with real-valued messages. However, we note that the training of FCMNet with binarized messages is more unstable, even if the general performance is improved with training. This is likely because the binarization processing interrupts the gradient propagation between agents, and a large quantity of information is lost during the conversion and de-conversion processes. They both raise the difficulty of learning collaboration between agents.

5.2.3 Lost Messages

In wireless digital communication, many factors can cause information loss, including noise interference, transmission delay, occlusions, device damage, etc. We believe that the performance of FCMNet under random message loss is crucial towards practical applications. In these experiments, we assume that

agents have a relatively low, fixed probability of losing a whole message at every message transmission, which is subsequently substituted by a zero vector of the same length. We actually selected the use of a zero vector as replacement message, because it performed best through a large set of control experiments that involved replacing the lost messages with the output hidden state of the self-memory unit, or a repeated message from the last time step. These control experiments likely suggest that relying on an outdated, and thus more likely incorrect, message may essentially introduce further noise that can more easily misguide agents. Conversely, we believe that using a constant zero-vector may explicitly signal the agent that a message was lost, allowing it to (learn to) react accordingly.

In order to determine the model collapse threshold, we ran multiple tests with varying message loss probabilities in the multi-agent pathfinding task. The primary results are presented in Figure 7b. These demonstrate that when the probability of message loss is less than or equal to 3%, the final converged episode length and converge speed are unaffected and equivalent to the standard FCMNet model without message loss. However, when the probability of message loss exceeds 3.5%, FCMNet stops working entirely. In this second scenario, both the actor and critic losses remain volatile and cannot be reduced via training. Therefore, we conclude that FCMNet can naturally resist (relatively low rates of) random message loss, where the probability threshold for this task is between 3%-3.5%.

5.2.4 Random Communication Order

In prior experiments, the topology of FCMNet was fixed, meaning that agents always transmit messages along each communication channel in a predetermined, fixed sequence. However, we believe this fixed sequence may not be flexible enough for practical applications. In this experiment, we further explore the performance of FCMNet when the transmission order of messages varies randomly at every timestep.

The evaluation curves of standard FCMNet and FCMNet with random connection order are shown in Figure 7c. Our results indicate that FCMNet with random connection order not only converges to the same performance level as standard FCMNet but also exhibits faster convergence in the multi-agent pathfinding task. We believe that this increased training speed may be due to the fact that randomizing the communication order increases the diversity of messages in the communication channel and makes communications richer, thus allowing agents to explore their cooperative policy space more extensively.

6 Conclusion

Learning team-level cooperation can be particularly challenging in partially observable multi-agent environments, where each agent's knowledge about the system is limited and the environment is unstable from the perspective of a single agent. To mitigate these risks, in this work, we proposed two RL models

that simultaneously learn a global communication mechanism to increase the information available to individual agents and a decentralized policy condition upon the message received. The first model, named FCMNet, learns a communication mechanism where agents sequentially transmit and encode the current observation-based message sent by every other agent at each timestep through multiple directed LSTM chains. Our second model, FCMTran, relies on the encoder of a modified transformer to parallely weight fuse self-generated messages sent by all agents at the previous timestep. In our results on the SMAC tasks with shared rewards, FCMNet and FCMTran both outperform recent communication-based methods and value decomposition methods in almost all tasks tested. Our models' performance can be further improved by combining them with existing value decomposition techniques (here, VDN and QMIX). The FCMTran variant combined with value decomposition even achieved strongly-above SOTA performance without task-specific tricks in one of the hardest SMAC tasks. Additionally, we further investigated the robustness of FCMNet in a multi-agent pathfinding task under three realistic communication disturbances (i.e., binarized messages, (relatively low) random message loss, and random communication order). Our work has been mostly simulation-based at this stage, and our future work will focus on deploying the two models to practical, robotics applications under realistic communication constraints.

Supplementary information. The supplementary material file of this paper contains the following videos: 1.) Fully trained FCMNet agents win the $5m\_vs\_6m$ task by the "focus-fire" skill (kill enemies in sequence). 2.) Fully trained FCMNet agents win the $2m\_vs\_1z$ task by avoiding the limited range of enemy's attacks. There, agents exhibit advanced cooperative behavior, i.e., one of the agents has learned to attract the enemy's fire through continuous back-and-forth movements, allowing another agent to safely attack the enemy (i.e., kiting behavior). 3.) Fully trained FCMNet agents win the $3s\_vs\_3z$ task by first dividing/scattering the enemy team and then eliminating them in turn. 4.) Fully trained FCMNet agents reach their target quickly, even if they do not know the exact location of the targets and the targets' location changes randomly.

Acknowledgments. We would like to thank Mehul Damani and Benjamin Freed for their feedback on earlier drafts of this paper. We are also grateful to Benjamin Freed and Rohan James for very helpful research discussions.

Declarations

- Funding: This work was founded by the Singapore Ministry of Education Academic Research Fund Tier 1.
- Conflict of interest/Competing interests: The authors have no relevant conflict of interest/competing interests to disclose.
- Ethics approval: Not applicable.
- Consent to participate: Not applicable.
- Consent for publication: All authors approved the paper to be published;

Table B1: Hyperparameters table

Hyperparameter	Value
Learning rate of policy net	3e-4 (1e-4 for $3m\_vs\_4s$ task)
Learning rate of value net	3e-4 (1e-4 for $3m\_vs\_4s$ task)
Discount factor	0.99
Gae lamda	0.95
Clip parameter for probability ratio ϵ	0.2
Gradient clip norm	20
Number of epoch	10
Number of processes	16
Mini batch size	512
Optimizer	AdamOptimizer
Entropy coefficient	0.01
Dimension of the LSTM cell in policy net and value net	64
Dimension of the feed-forward network of the encoder	1024
Number of heads	8
Dimension of q,k,v	32
Dimension of the mixing network hidden layer (per agent)	64

- Availability of data and materials: Not applicable.
- Code availability: Code will be made available publicly upon paper acceptance.
- Authors' contributions: Yutong Wang and Guillaume Sartoretti contributed to the study conception and design. Code writing, data collection and analysis were performed by Yutong Wang and Yizhuo Wang. The first draft of the manuscript was written by Yutong Wang and all authors then commented and edited the final manuscript.

Appendix A Experimental Setup

For both FCMNet and FCMTran, the code of the neural network part was written in torch 1.9.0 and relied on Ray 1.2.0 to employ 16 processes to collect data in parallel. The convergence speed of models varies with different tasks. For the $5m\_vs\_6m$ task, on a computer equipped with one Nvidia GeForce RTX 2080 Ti GPUs and one Intel(R) Core(TM) i9-10900KF CPU (10 cores, 20 threads), using the standard hyperparameters listed in next section, FCMNet converges within 3M timesteps (1.3 hours of wall clock time), FCMTran will converge within 20M timesteps (6.5 hours).

Appendix B Hyperparameters

Table B1 presents the hyperparameters used to train our models evaluated in Section 5.1 of this paper. These hyperparameters were obtained by coarse and empirical hyperparameter tuning without detailed grid searches (i.e., we believe that it is not fair to compare our models with fine-tuned QMIX), and

can be used for every SMAC task listed in the paper. The first 11 hyperparameters are general hyperparameters for training neural networks using PPO, 12-15 are unique hyperparameters for FCMNet and FCMTran, and the last hyperparameter is for value decomposition.

References

[1] Arulkumaran, K., Cully, A., Togelius, J.: Alphastar: An evolutionary computation perspective. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion, pp. 314–315 (2019)

[2] Berner, C., Brockman, G., Chan, B., Cheung, V., Debiak, P., Dennison, C., Farhi, D., Fischer, Q., Hashme, S., Hesse, C., et al.: Dota 2 with large scale deep reinforcement learning. arXiv preprint arXiv:1912.06680 (2019)

[3] Kiran, B.R., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A.A., Yogamani, S., Pérez, P.: Deep reinforcement learning for autonomous driving: A survey. IEEE Transactions on Intelligent Transportation Systems (2021)

[4] Wang, S.-J., Chang, S.: Autonomous bus fleet control using multiagent reinforcement learning. Journal of Advanced Transportation **2021** (2021)

[5] Damani, M., Luo, Z., Wenzel, E., Sartoretti, G.: Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. IEEE Robotics and Automation Letters **6**(2), 2666–2673 (2021)

[6] Sartoretti, G., Wu, Y., Paivine, W., Kumar, T.S., Koenig, S., Choset, H.: Distributed reinforcement learning for multi-robot decentralized collective construction. In: Distributed Autonomous Robotic Systems (DARS 2018), pp. 35–49 (2019)

[7] Wang, Y., Damani, M., Wang, P., Cao, Y., Sartoretti, G.: Distributed reinforcement learning for robot teams: a review. Current Robotics Reports **3**(4), 239–257 (2022)

[8] Hernandez-Leal, P., Kartal, B., Taylor, M.E.: Is multiagent deep reinforcement learning the answer or the question? a brief survey. learning **21**, 22 (2018)

[9] Kim, D., Moon, S., Hostallero, D., Kang, W.J., Lee, T., Son, K., Yi, Y.: Learning to schedule communication in multi-agent reinforcement learning. arXiv preprint arXiv:1902.01554 (2019)

[10] Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., Gao, Y.: Multi-agent game abstraction via graph attention neural network. In: Proceedings of the

AAAI Conference on Artificial Intelligence, vol. 34, pp. 7211–7218 (2020)

[11] Jiang, J., Lu, Z.: Learning attentional communication for multi-agent cooperation. Advances in neural information processing systems **31** (2018)

[12] Samvelyan, M., Rashid, T., De Witt, C.S., Farquhar, G., Nardelli, N., Rudner, T.G., Hung, C.-M., Torr, P.H., Foerster, J., Whiteson, S.: The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043 (2019)

[13] Rashid, T., Samvelyan, M., Schroeder, C., Farquhar, G., Foerster, J., Whiteson, S.: Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: International Conference on Machine Learning, pp. 4295–4304 (2018). PMLR

[14] 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 preprint arXiv:1706.05296 (2017)

[15] Freed, B., Sartoretti, G., Hu, J., Choset, H.: Communication learning via backpropagation in discrete channels with unknown noise. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, pp. 7160–7168 (2020)

[16] Foerster, J., Assael, I.A., De Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. Advances in neural information processing systems **29** (2016)

[17] Sukhbaatar, S., Fergus, R., et al.: Learning multiagent communication with backpropagation. Advances in neural information processing systems **29** (2016)

[18] Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., Wang, J.: Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. arXiv preprint arXiv:1703.10069 (2017)

[19] Kong, X., Xin, B., Liu, F., Wang, Y.: Revisiting the master-slave architecture in multi-agent deep reinforcement learning. arXiv preprint arXiv:1712.07305 (2017)

[20] Niu, Y., Paleja, R.R., Gombolay, M.C.: Multi-agent graph-attention communication and teaming. In: AAMAS, pp. 964–973 (2021)

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in

neural information processing systems **30** (2017)

[22] Li, W., Luo, H., Lin, Z., Zhang, C., Lu, Z., Ye, D.: A survey on transformers in reinforcement learning. arXiv preprint arXiv:2301.03044 (2023)

[23] Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R.L., Clark, A., Noury, S., *et al.*: Stabilizing transformers for reinforcement learning. In: International Conference on Machine Learning, pp. 7487–7498 (2020). PMLR

[24] Cao, Y., Wang, Y., Vashisth, A., Fan, H., Sartoretti, G.A.: CAt-NIPP: Context-aware attention-based network for informative path planning. In: 6th Annual Conference on Robot Learning (2022). https://openreview.net/forum?id=cAIIbdNAeNa

[25] Cao, Y., Hou, T., Wang, Y., Yi, X., Sartoretti, G.: Ariadne: A reinforcement learning approach using attention-based deep networks for exploration. arXiv preprint arXiv:2301.11575 (2023)

[26] Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., Mordatch, I.: Decision transformer: Reinforcement learning via sequence modeling. Advances in neural information processing systems **34**, 15084–15097 (2021)

[27] Shang, J., Kahatapitiya, K., Li, X., Ryoo, M.S.: Starformer: Transformer with state-action-reward representations for visual reinforcement learning. In: European Conference on Computer Vision, pp. 462–479 (2022). Springer

[28] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)

[29] 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, pp. 770–778 (2016)

[30] Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)

[31] Su, J., Adams, S., Beling, P.: Value-decomposition multi-agent actor-critics. Proceedings of the AAAI Conference on Artificial Intelligence **35**(13), 11352–11360 (2021). https://doi.org/10.1609/aaai.v35i13.17353

[32] Yu, C., Velu, A., Vinitsky, E., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of ppo in cooperative, multi-agent games. arXiv preprint arXiv:2103.01955 (2021)

[33] Hu, J., Jiang, S., Harding, S.A., Wu, H., Liao, S.-w.: Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2102.03479 (2021)

[34] 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 based on team reward. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, pp. 2085–2087 (2018)

[35] Courbariaux, M., Bengio, Y., David, J.-P.: Binaryconnect: Training deep neural networks with binary weights during propagations. In: Advances in Neural Information Processing Systems, pp. 3123–3131 (2015)

[36] Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**(3), 229–256 (1992)

[37] Toderici, G., O'Malley, S.M., Hwang, S.J., Vincent, D., Minnen, D., Baluja, S., Covell, M., Sukthankar, R.: Variable rate image compression with recurrent neural networks. arXiv preprint arXiv:1511.06085 (2015)